UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

PERFORMANCE ENHANCEMENT OF LARGE SCALE NETWORKS WITH HETEROGENEOUS TRAFFIC

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements of the

degree of

Doctor of Philosophy

By

MOSTAFA HASSAN DAHSHAN
Norman, Oklahoma
2006

PERFORMANCE ENHANCEMENT OF LARGE SCALE NETWORKS WITH HETEROGENEOUS TRAFFIC

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Pramode K. Verma

Dr. James J. Sluss, Jr.

Dr. Stamatios V. Kartalopoulos

Dr. John Y. Cheung

Dr. William O. Ray

Acknowledgements

I would like to express my gratitude to my advisor, Dr. Pramode Verma for the guidance and support he provided to me during my Masters and PhD studies. I also want to thank my respected committee members, Dr. James Sluss, Dr. Stamatios Kartalopoulos, Dr. John Cheung and Dr. William Ray, for their comments and recommendations to improve this dissertation.

Table of Contents

List of Illustrations

List of Tables

List of Publications

Conferences with Proceedings

- **Mostafa H. Dahshan** and Pramode K. Verma, "Pricing for Quality of Service in High Speed Packet Switched Networks," IEEE Workshop on High Performance Switching and Routing – HPSR 2006, Poznan, Poland, June 7-9, 2006.
- **Mostafa H. Dahshan** and Pramode K. Verma, "Performance Enhancement by Segregation and Hybrid Integration in General Queueing Networks," International Symposium on Performance Evaluation of Computer and Telecommunication Systems – SPECTS 2005, Philadelphia, PA, July 24-28, 2005, pp 143-148.
- **Mostafa H. Dahshan** and Pramode K. Verma, "Performance Enhancement of Heavy Tailed Queueing Systems using a Hybrid Integration Approach," IEEE Global Telecommunications Conference – GLOBECOM 2005, St Louis, MO, November 28-December 2, 2005.
- Sarma V. Nedunuri, **Mostafa H. Dahshan** and John Y. Cheung, "A Dynamically Reconfigurable, Decentralized Multiparty Teleconferencing System Over IP," Proceedings of the 15th International Conference on Parallel and Distributed Computer Systems, pp.136-140, September, 2002.

Journals

- **Mostafa H. Dahshan** and Pramode K. Verma, "Refined Assured Forwarding Framework for Differentiated Services Architecture," Submitted, IEE Proceedings Communications, 2006.

Abstract

This dissertation provides novel techniques for improving the Quality of Service by enhancing the performance of queue management in large scale packet switched networks with a high volume of traffic. Networks combine traffic from multiple sources which have disparate characteristics. Multiplexing such heterogeneous traffic usually results in adverse effects on the overall performance of the network.

This dissertation builds on the notion that segregating traffic with disparate characteristics into separate channels generally results in a better performance. Through a quantitative analysis, it precisely defines the number of classes and the allocation of traffic into these classes that will lead to optimal performance from a latency standpoint. Additionally, it weakens the most generally used assumption of exponential or geometric distribution of traffic service time in the integration versus segregation studies to date by including self-similarity in network traffic.

The dissertation also develops a pricing model based on resource usage in a system with segregated channels. Based on analytical results, this dissertation proposes a scheme whereby a service provider can develop compensatory and fair prices for customers with varying QoS requirements under a wide variety of ambient traffic scenarios.

Finally, these findings are applied towards improving the performance of the Differentiated Services architecture by developing a new Refined Assured Forwarding framework where heterogeneous traffic flows share the same aggregate

class. The new framework requires minimal modification to the existing Diffserv routers. The efficiency of the new architecture in enhancing the performance of Diffserv is demonstrated by simulation results under different traffic scenarios.

Chapter 1. Introduction

Abstract: This chapter reviews switching techniques and the parameters used to characterize communication networks performance. It discusses the need for implementing Quality of Service techniques in communications networks. The chapter concludes by identifying the scope and contributions of this dissertation in packet switching, especially in the context of Quality of Service parameters for communication networks.

Telecommunications services enable communication between sources and destinations. Telecommunication services are generally offered through a common user network. Users demand services when they need them and expect to be provided with the service they need with an acceptable level of price and performance.

Common user networks are designed to meet their performance objectives on a statistical basis, rather than on a guaranteed performance basis. The cost of providing connectivity among any source-destination pair at all times would be prohibitively high. Therefore, connectivity is provided on a switched or on-demand basis, plus all such connectivity are provided with entry statistical and not absolute guarantees. This dissertation addresses the general problem of resource optimization in packet switched networks while providing fairness among multiple classes of traffic. Resource optimization is an important goal for network providers. It enables higher utilization of resources, better service quality and lower operational cost.

1.1 Switching Techniques

Switching techniques can be divided into two main categories: circuit switching and packet switching. In circuit switching, the end-to-end path must be established prior to data transfer. If the end-to-end path cannot be established, the connection is blocked. All resources along the path are dedicated for the connection until it is terminated. Both ends of a circuit switched connection must in general communicate at the same transfer rate [1]. Circuit switching was originally designed for voice traffic, but it has also been used for data networks, such as ISDN. Today, circuit switching is used in high speed optical networks such as Synchronous Optical Networks (SONET) using the Dense Wave Division Multiplexing (DWDM) [2].

In packet switching, the information to be transmitted is divided into packets of small sizes (typically, a few hundreds of bytes). Packets are transmitted from node to node across the network until they reach their destination. Generally, packets need not follow the same path. Routing decision is made at each node in order to determine the next node on the path to destination. Thus, packet switched networks are part of the class of store and forward networks [3].

Packet switching uses either datagram or virtual circuit. The datagram approach is connectionless. Packets are transmitted as needed, without any prior communication to the receiving node. Each packet might follow a different path, and therefore the successive packets are not guaranteed to be delivered in order. It is the responsibility of the receiving node to reorder received packets. The virtual circuit approach is connection oriented. Similar to circuit switching, the end to end path must be established before data transfer can take place. Once the path is established, all

packets follow the same path. No reordering of packets is needed. However, unlike the circuit switching, resources along the path are not dedicated to a single virtual circuit connection. Multiple virtual circuits can share the same link, provided that the link capacity is sufficient to accommodate all simultaneous connections.

The circuit switching approach has the following advantages:

- Resources availability is guaranteed. Once the channel has been allocated upon connection establishment, it is not shared by other connections.
- Since all transmitted data follow the same path, there is no overhead added to packets to indicate their destination.
- No processing overhead is incurred to determine the next hop along the path since it has already been determined during the connection establishment process.
- Virtual circuits share most of the advantages of circuit switching with the exception of the guaranteed resource availability.

The disadvantages of the circuit switching approach include the following:

- Utilization of resources is low, particularly when it is used in data communications. The line might be idle most of the time while resources remain dedicated for the connection [1]. This is particularly true for bursty traffic.
- Connections are blocked when the end to end path cannot be established.
- Data is transmitted over a single path, which make the system prone to failures. A single failure at any point along the path will result in communication interruption.

Packet switching provides better resource utilization since resources are not dedicated to a single connection during the connection lifetime. In datagram-based

networks, packets may follow different paths. This provides some redundancy against network failures. Unlike circuit switching, connections are not blocked when the network is congested. The network will always accept packets, but packet delay or loss might result, depending on the buffer space available in switching nodes [1].

The Internet is a typical example of a packet switched network. Although circuit switching is used in the Internet core, this switching is semi static and is transparent to the routing protocols [4]. Circuit switched connections at the Internet backbone can be simply considered as logical point to point links.

The Internet Protocol (IP) used in the Internet was designed as a best effort protocol. Best effort means that the network does not guarantee any particular level of performance nor does it guarantee the delivery of packets. Routing protocols determine the best available paths. Packets are transmitted from node to node "hoping" that they will reach their destination in reasonable time. It is the responsibility of upper layer protocols or applications to implement their own mechanisms to ensure the integrity of transmitted data [5]. However, higher layer protocols cannot provide any guarantees for other performance parameters, such as delay, jitter or any special treatment for certain types of traffic.

1.2 The Need for Quality of Service

For many years, the Internet was able to satisfy most of the user requirements such as email, remote access, and file transfers. With the widespread availability of the World Wide Web, as well as the growth of the Internet usage, user demands have increased substantially and resource demands of Internet applications became more diverse [6, 7].

Best effort service reduces the complexity of network devices, enabling faster routing and switching decisions. On the other hand, the diversity of network applications, as well as user demands, makes the best effort model inadequate. Different applications have different requirements. For example, real time voice and video applications are delay sensitive but can tolerate some data loss. Voice doesn't require high bandwidth while video does. Video streaming, on the other hand, requires high bandwidth but can be buffered and so can tolerate some delay. File transfer and data applications cannot tolerate packet loss, and so on [6].

As a result, techniques beyond best effort have become essential. These techniques are collectively referred to as Quality of Service (QoS) techniques. The purpose of QoS is to provide different levels of service to different classes of traffic. The class can be an application type, traffic from a certain user, specific flow, etc.

In addition to the diversity of network applications and the increasing computing power, other factors have contributed to the need for quality of service:

- The deregulation and privatization of the telecommunication industry in many countries, including the U.S., have made it inevitable for service providers to improve the quality of their networks to gain competitive advantage [8].
- Offering different levels of services allows accommodation of broader range of customers. Lower level services can be offered at lower prices for budget customers and higher service levels can be made available for customers who need them. Different users have different requirements and willingness to pay.
- Even if no pricing differentiation is offered, service differentiation is needed to provide different applications with services that they need. For example, some

TCP applications can tolerate some delay, but not loss. On the other hand, Some UDP applications such as voice will prefer their packets to be discarded rather than being delayed beyond a certain limit [5].

- Virtual Private Networks (VPNs) are widely used today to provide connectivity between corporate locations spanning large geographic areas. Traffic from different VPNs typically needs to be isolated form other traffic [5, 9].
- Many broadband Internet users are now switching from regular phone lines to voice over IP (VoIP) phones. As such, it is essential that QoS techniques be implemented to differentiate voice traffic from other traffic, particularly for broadband service providers who provide their own VoIP services, such as Cox, Verizon and AOL.

1.3 Quality of Service Performance Parameters

Quality of Service can be quantified using a wide range of performance parameters. These parameters can in general be different from the end user's perspective and the network provider's perspective. Service providers should ensure their evaluation of the network's Quality of Service is related to how users would rate the service and map them to the actual performance parameters [8]. Performance parameters fall into the following categories [3, 6]:

1.3.1 Timeliness related parameters

Timeliness parameters include delay, response time and jitter. Delay is the time it takes the packet to reach its destination. It is composed of transmission delay, queueing delay, processing delay and propagation delay [10]. Transmission delay

depends on the line speed in bits per second (bps), also referred to as channel capacity. The higher the speed the channel can transfer data, the lower the transmission delay. Queueing delay is incurred when packets are arriving faster than the switching device can forward them. Packets are queued in the device buffer until they can be transmitted. Processing delay is the time required for the switching device to process packet headers, and make the switching or routing decision accordingly. Propagation delay is the signal propagation time and is dependent on the transmission medium (copper, wireless channel, optical fiber, etc).

Response time is a higher level parameter that is relevant in connection oriented protocols such as TCP over IP. In these protocols, each packet or set of packets must be acknowledged by the destination node. The time it takes the packet to be transmitted and acknowledged back is the response time.

Jitter is the variation in the delay. It is a critical parameter in real-time applications such as VoIP and video conferencing. The jitter effect is usually caused because of the changing network congestion status which causes subsequent packets within the same flow to encounter different delay values. For real time applications, this causes an annoying effect. Jitter is usually circumvented by using playback buffers, where packets arriving at a variable rate are forwarded at a constant rate. The larger the jitter value, the larger the buffer size required. For real-time applications, buffer sizes should be limited to accommodate the duration for which received packets are still useful [5, 11].

1.3.2 Throughput related parameters

Throughput is the amount of data received per unit time. Throughput requirements depend on the application. While some applications can be flexible in their throughput requirements, other applications such as multimedia streaming and video conferencing require a minimum amount of guaranteed throughput [11].

1.3.3 Integrity related parameters

Integrity parameters measure the accuracy of data transmission. Transmission errors can result in corruption of packet headers or payloads. Errors in packet headers can result in undelivered packets. Depending on the application, errors in payload may render packets useless when received. Integrity parameters include packet loss ratio and bit error rate (BER). Packet loss ratio is the percentage of the number of packets lost (or delayed beyond the maximum allowed time) to the total number packets transmitted. The bit error rate is the probability of a bit being corrupted during transmission. Packet loss can occur due to congestion, node failure or link failure. Bit error rate is mainly dependent on the physical medium, signal to noise ratio and surrounding interference. Bit error rate can be countered by data encoding techniques. Some of these techniques can reduce the effective throughput by using part of the bandwidth to provide some redundancy [1]. Packet loss caused by congestion can be handled by efficient queue management techniques, as studied in this dissertation.

1.3.4 Availability related parameters

Availability parameters measure the percentage of time network resources are available. They include Mean Time To Failure (MTTF), Mean Time To Repair (MTTR), Mean Time Between Failures (MTBF) and Percentage of time available [6]. Availability parameters and integrity parameters together are sometimes referred to as Reliability parameters.

1.3.5 Fairness

Telecommunication networks are shared resources. They are used by multiple users simultaneously. This raises the issue of fairness between and among these users. At the application level, TCP applications tend to back off when packet loss occurs signaling a congestion while UDP applications keep pushing more traffic because UDP doesn't implement a mechanism for congestion control. This can result in unfairness at the application level. A similar case arises when an application with large packet sizes or bandwidth requirement shares the same channel with another application with smaller packet sizes or bandwidth. On networks were priorities are implemented, this issue is even more complicated, as fairness problems occur between different classes or within the same class or both [12]. Unlike other QoS parameters, there is no well defined measure of fairness [13]. In this dissertation, the system will be considered fair if flows or packets that have the same priority receive equal share of resources, such as bandwidth and queue space. At the time of congestion, the delay and packet loss should be evenly distributed among different flows.

1.4 Traffic Characterization

An important factor in designing efficient QoS techniques is understanding the behavior of network traffic. For a long time, the general perception was that voice and data traffic obey certain Markovian stochastic models, particularly the Poisson process. These models treat packet arrival as an independent (or short range dependent) and memoryless process. Occasionally, some bursts of condensed arrivals may happen, but these variations tend to smooth over larger time scales. The same can be said about packet sizes. Traffic behavior over larger time scales is a determining factor in the design of both buffer space and bandwidth allocation requirements. The Poisson models enabled accurate and tractable mathematical treatment. They have, however, turned out to be optimistic design models for QoS requirements in the past two decades [14].

Traffic characterization has fundamentally changed with the discovery of the self-similar nature of network traffic in the early 1990s [15]. Self-similarity, in the context of network traffic, means that correlations between packet arrivals or packet sizes are persistent over wide range of time scales. Poisson models suggest that the probability of large bursts of packets or very long packets decreases exponentially. In reality, those “unusual” or “unwanted” events occur with non-negligible probability.

Different approaches are being implemented for characterizing the self-similar behavior of network traffic. Research work in this area generally falls under one of the categories discussed in the following subsections [14].

1.4.1 Measurement Based Modeling

In measurement based modeling, the self similarity or long range dependence of network traffic are identified by analyzing traffic traces or logs from actual networks such as corporate local area networks or network servers. The outcome of the analysis is compared with different mathematical models. One problem with this approach is that several models can closely match the traffic traces, which makes it difficult to select a particular model [14].

1.4.2 Physical Modeling

Physical modeling attempts to physically rationalize the self-similar behavior of network traffic based on the interaction between multiplexed traffic flows or relate this behavior to the characteristics of network applications which originate those flows. Some studies claim that self-similarity is caused by the arrival patterns of certain applications such as multimedia video streaming, which suggests that self-similarity is inherent in these applications. Such hypothesis, however, does not explain the observation of self-similarity on traffic traces collected during a time period that preceded the existence of video applications. Other studies justify self-similarity by the fact that the distribution of file sizes on network servers is heavy-tailed, which means that the probability of the existence of large files is non-negligible. Transferring heavy-tailed distributed files, according to those studies, results in self-similar traffic across the network [14].

1.4.3 Queueing Analysis

Research work in this category aims to provide the mathematical models that capture traffic characteristics for the purpose of performance analysis. These models are described in Chapter 3. Some of these models are utilized in this dissertation.

1.4.4 Traffic Control and Resource Provisioning

Self-similarity has a profound effect on resource requirements. It also imposes more challenges on the task of network design and traffic control. The research in this category aims to study the resource requirements and utilization in terms of bandwidth and buffer space. Other important aspects of this category are the effects of resource sharing between multiple flows on network performance and resource utilization. The work of this dissertation partially falls under this category.

1.5 Scope and Contribution of the Dissertation

Delay in packet switched networks is possibly the most important parameter of importance from the user's standpoint. Accordingly, minimization of delay is an important objective of the network designer. This dissertation focuses on datagram-based packet switched networks, since this is the model implemented in the Internet. The goal of this dissertation is to improve the delay and the fairness parameters by optimizing queue management and handling. Two types of networks are considered. The first type is best effort networks, which are the most widely available in today's Internet. The second type is QoS enabled networks, particularly implementing the Differentiated Services architecture.

For best effort networks, the dissertation proposes segregation and hybrid integration of traffic with similar characteristics as the two approaches for lowering delay in point-to-point links, which will eventually lower the end-to-end delay as well. Segregation means allocating a separate queue for each flow of traffic based on defined criteria. Segregation is contrasted with integration, where all flows of traffic share the full available bandwidth. Hybrid integration is presented as a more intelligent and dynamic approach that combines the advantages of both segregation and integration at the cost of some additional processing overhead in switching devices. In proving the merits of the segregated and hybrid approaches, the self-similar nature of the Internet traffic is taken into consideration. This entails the utilization of more complicated mathematical models than those used in classical queueing analysis. Detailed mathematical analysis is provided and supported by simulation results.

For QoS enabled networks, this dissertation includes two important studies. The first study is to understand the cost associated with exclusive resource allocation. A pricing model is proposed that is proportional to the actual bandwidth usage in Integrated Services networks. The second study introduces a new approach utilizing the concept of hybrid integration, termed the Refined Assured Forwarding framework, for Differentiated Services Architecture. The new approach significantly improves the performance of Differentiated Services networks in terms of delay, throughput and packet loss, in addition to fairness.

1.6 Organization of the Dissertation

This dissertation consists of two main parts. The first part (Chapters 2 to 5) deals with best effort networks. Chapter 2 provides a basic background on the queueing analysis including notations and equations used in determining the packet delay. Chapter 3 provides an overview of self-similarity in Internet traffic and presents a mathematical framework for delay analysis using self-similar traffic models. Chapter 4 illustrates the segregated and integrated approaches and provides the analytical framework for calculating the delay under each approach. The proposed hybrid integrated approach is presented in detail in Chapter 5. Detailed analytical and simulation results are provided that demonstrate its performance improvement.

The second part of the dissertation (Chapters 6 to 8) focuses on networks deploying Quality of Service. Chapter 6 reviews the current QoS techniques, namely IntServ, DiffServ and MPLS comparing their strengths and weaknesses. Chapter 7 addresses the impact of Quality of Service on bandwidth requirements and proposes a scheme whereby a service provider can develop compensatory and fair pricing for customers with varying QoS under a wide variety of ambient traffic scenarios. Chapter 8 presents a new framework termed the Refined Assured Forwarding (RAF) framework for improving the performance of DiffServ architecture where heterogeneous traffic flows share the same aggregate class. The new framework requires minimal modification to existing DiffServ routers. The efficiency of the new architecture in enhancing the performance of DiffServ is demonstrated by simulation results under different traffic scenarios. Chapter 9 presents a summary of the major results and the conclusions of this dissertation.

Chapter 2. Queueing Theory

Abstract: This chapter provides an overview of the queueing theory. It describes the terminology and notations used in queueing analysis. This chapter focuses on delay analysis in packet switched networks with classical queueing models and Poisson packet arrivals. Self-similar traffic analysis is presented in Chapter 3.

Queues are encountered in many situations in daily life. Some examples include water behind a dam, cars waiting at traffic lights and customers at checkout counters. Queueing systems are a subclass of flow systems. Flow systems are systems in which objects are transferred from one point to another through channels of finite capacities. Flows are classified into steady flows and unsteady flows. In steady flows, objects arrive at a predictable or constant rate and the demand or service time of each object is known. In unsteady flows, both objects arrival pattern and the demand of each object are random. Unsteady flows are also known as stochastic flows. The flow system is considered to be stable if the channel capacity is larger than the average arrival rate. In stochastic flows, however, queues may still be formed, with finite length, when the instantaneous object arrival rate is higher the channel capacity. If the average arrival rate itself is larger than the channel capacity, the queue grows without bound and the system is considered unstable [16]. The classification of flow types is shown in Figure 2.1.

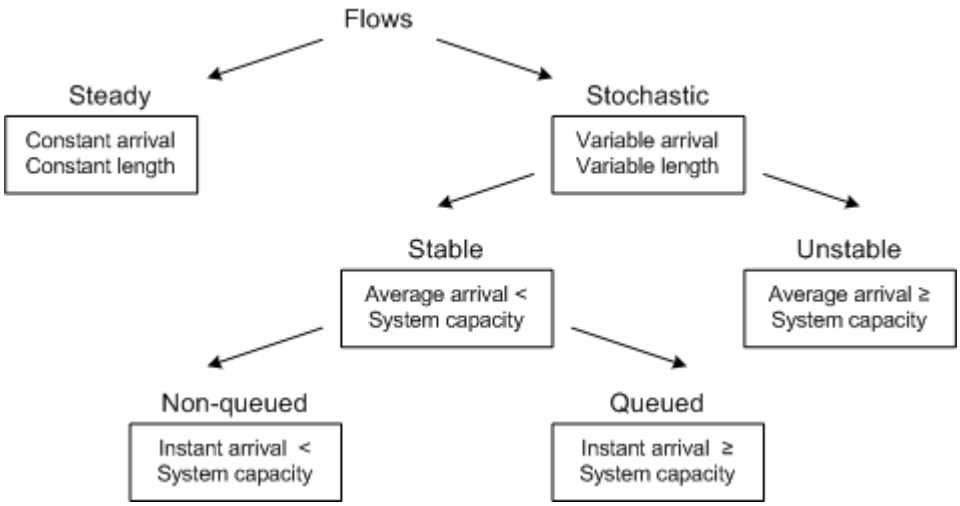

Figure 2.1: Flow types

Packet switched networks are an example of stochastic flow systems. In packet switched networks, channels are the network links and the flow objects are packets. The entity that serves the packets is typically a network device such as a router or a switch. In this dissertation, the purpose of studying queueing theory is to improve the delay caused by queues. As mentioned in Chapter 1, packet delay is composed of four components: transmission delay, queueing delay, processing delay and propagation delay. In the analysis performed in this dissertation, as in most similar studies, the processing and propagation delays are ignored and considered negligible compared to the queueing and transmission delays. This dissertation focuses on the queueing part of the delay.

2.1 Specification of Queueing Systems

Queueing analysis is based on stochastic processes. In its basic form, a queueing system is described by the distributions of stochastic processes of the time between arrivals of packets (interarrival time) and the time it takes each packet to be

serviced (service time). In a more advanced form, some additional parameters are used including the buffer capacity, the population of arriving packets and the queueing discipline [16].

A special notation called Kendall's notation has been defined to describe queue types in terms of the interarrival times, service times and other parameters. The basic Kendall notation is in the format: *X*/*Y*/*Z*. *X* represents the distribution of the interarrival time process. *Y* represents the distribution of the service time process, which is directly related to the packet size. *Z* is the number of service facilities (switching nodes) [11]. A more advanced notation is in the format: *X*/*Y*/*Z*/*A*/*B*, where *A* is the buffer capacity and *B* is the population of arriving packets [16]. Table 2.1 shows symbols commonly used to represent certain types of distributions.

Letter	**Meaning**
G	General distribution of service times or interarrival times
M	Exponential (Markovian) distribution.
D	Deterministic or fixed service times ore interarrivals

Table 2.1: Common letters used in Kendall's notation

2.2 Queue Distributions

Analytical modeling of queueing analysis is generally a complex task. The complexity varies depending on the distributions of interarrival and service times, as well as the assumptions about buffer sizes and the arrival population. This section describes some of the queue types used in this dissertation. To simplify the analysis, the following assumptions will be made:

- The buffer size is assumed to be infinite. Effectively this means that buffers are large enough to sustain transient congestions.
- The arrival population is assumed infinite.

- Only point-to-point links are considered. i.e., the number of servers for each queue is 1.

In summary, these assumptions mean that queue types used in this dissertation are fully described by the basic Kendall notation $X/Y/Z$ with Z always equal to 1.

2.2.1 M/M/1 Queues

The M/M/1 queue is the simplest queue system. In this system, the interarrival time is exponentially distributed. Equivalently, the arrival process is a Poisson process. Service times are also exponentially distributed. Because service time is directly related to packet size, this also means that packet sizes are also exponentially distributed. Poisson arrivals mean that packet arrivals are random and independent.

The M/M/1 queue system has long been used in queueing analysis because of the simplicity of its mathematical model. The Poisson or random nature of packet arrivals has been justified by the fact that variations in traffic coming from different sources are smoothed when traffic is aggregated or multiplexed. This becomes more apparent with increasing network load [17]. Exponential service times imply that service times or packet sizes are varying with a random nature. Typical examples are terminal to computer communications, airline reservations and shared Local Area Networks [11].

2.2.2 M/D/1 Queues

M/D/1 queues are similar to M/M/1 queues except that service times are constant. It is suitable for networks with fixed packet sizes, such as ATM networks.

2.2.3 M/G/1 Queues

Both M/M/1 and M/D/1 queues are two special cases of the broader M/G/1 queue category. M/G/1 queues have arbitrary distribution of service time. Since the service time distribution doesn't necessarily possess the Markovian/memoryless property, characterization of M/G/1 queues must take the time factor into consideration, which adds to the complexity of their mathematical treatment. The mathematical model of the M/G/1 queue is used in this dissertation to analyze queues with heavy-tailed distributions of service times.

2.2.4 G/G/1 Queues

G/G/1 queues are the very generic type of queues, where both interarrival and service times have arbitrary distributions. They involve very complex mathematical treatment. Analytical models for characterizing G/G/1 queues are generally approximate models, and they usually require some simplifying assumptions under which the approximations provide valid results [16]. It is imperative to utilize G/G/1 queue models in order to characterize self-similar network traffic. This dissertation makes use of some of the approximate G/G/1 queue models. However, under certain conditions, these models cannot be applied and it is necessary to utilize simulation tools to provide approximate results.

2.3 Average Packet Delay

Among the other parameters that characterize the performance of packet switched networks, the average delay is the most important parameter from both user and designer standpoints. Most of the work of this dissertation is devoted to improve

the average delay and particularly the queueing component of the delay. Table 2.2 summarizes the notations that are used in queueing analysis and delay calculations throughout this dissertation [11, 18]. Other notations will be defined as needed.

Symbol	Meaning	Explanation
C	Channel capacity or link speed	Maximum possible throughput in bytes per second
λ	Average arrival rate	The average number of packets entering the queue per unit time
$1/\mu$	Average packet size	Average number of bytes per packet
ρ	Utilization factor	The fraction of time the switching node is busy delivering the traffic on the egress link. It is equal to the ratio between the arrival rate and the service rate
σ_S^2	Service time variance	
σ_D^2	Delay variance	Variance of the delay is the jitter
σ_τ^2	Interarrival time variance	

Table 2.2: Notations used in this dissertation

An important relation that is used to determine the queueing delay is known as Little's result. It states that "the average number of customers in a queueing system is equal to the average arrival rate of customers to that system, times the average time spent in that system" [16]. Let N denote the average number of packets in the system, T denote the average packet delay. Using Little's result, N can be calculated as follows [16]:

$$N = \lambda T \tag{2.1}$$

Note that N is composed of the average number of packets waiting in the queue and the average number of packets being serviced. The latter quantity is can be shown to be equivalent to the utilization factor ρ, which is the fraction of time the server is busy [16]. Let N_q denote the average number of packets waiting in the queue but not being serviced, the following relation applies:

$$N = N_q + \rho \tag{2.2}$$

The average packet delay T is composed of the waiting time (queueing delay) $\overline{w}$ and the service time (transmission delay) $\overline{s}$. Thus:

$$T = \overline{w} + \overline{s} \tag{2.3}$$

An important property of Little's result is that it is independent of the distribution of the interarrival or service time. Thus, it can be utilized in different types of queues [16].

The rest of this section provides the specific relations used to determine the average number of packets in the system for the different queue types studied in this dissertation. The average packet delay is then calculated from Little's result.

2.3.1 M/G/1 Queues

For M/G/1 queues, the average number of packets in the system N is calculated as follows:

$$N = \frac{\rho^2 + \lambda^2 \sigma_s^2}{2(1-\rho)} + \rho \tag{2.4}$$

This important relation is known as the Pollaczek-Khintchine (P–K) mean value formula. It has a great importance in queueing theory [16, 19, 20]. Using this formula, the average packet delay for the M/G/1 queueing system can be calculated as follows:

$$T = \frac{\rho^2 + \lambda^2 \sigma_S^2}{2\lambda(1-\rho)} + \frac{1}{\mu C} \tag{2.5}$$

2.3.2 M/M/1 Queues

Since the M/M/1 queue is as special case of the M/G/1 queue with exponential service time, the average delay for the M/M/1 can be calculated using the P–K formula. First, the mean service time $\overline{s}$ is calculated as:

$$\overline{s} = \frac{1}{\mu C} \tag{2.6}$$

Because the service time is exponentially distributed, the variance σ_S^2 of the service time is calculated as:

$$\sigma_s^2 = \frac{1}{\left(\mu C\right)^2} \tag{2.7}$$

Substituting in (2.5) yields:

$$T = \frac{1}{\mu C - \lambda} \tag{2.8}$$

2.3.3 G/G/1 Queues

The average delay for G/G/1 queues can be approximated by different methods. A simple and powerful approximation is due to Kingman [21] and provides an upper bound on the average waiting time, as follows:

$$\overline{w} \le \frac{\sigma_\tau^2 + \sigma_s^2}{2\left(\overline{\tau} - \overline{s}\right)} \tag{2.9}$$

The average packet delay can then be calculated using (2.3):

$$T \le \frac{\sigma_\tau^2 + \sigma_s^2}{2\left(\overline{\tau} - \overline{s}\right)} + \overline{s} \tag{2.10}$$

The Kingman's bound provides more accurate results under heavy traffic load. i.e., as the utilization factor $\rho \to 1$. Kingman's bound does not depend on the

distributions of the interarrival or service time. However, it requires that the first and second moments of these distribution exist [22]. As will be mentioned later in this dissertation, some heavy-tailed distributions that are used to model self-similar traffic do not have finite values for the first and second moment. Kingman's bound cannot be used with those distributions.

2.4 Summary

This chapter has provided a brief overview of the queueing theory related to the work of this dissertation. The delay formulas for M/M/1, M/G/1 and G/G/1 have been discussed. In the following chapter, the self-similar nature of network traffic is illustrated in more detail, including the analytical models used to characterize it.

Chapter 3. Self-Similar Traffic

Abstract: This chapter highlights the self-similar nature of network traffic. It demonstrates the effects of self-similarity on network performance. Mathematical models used to characterize the self-similarity are also presented.

3.1 Self-Similarity

Self-similarity is the characteristic of the persistence of a certain pattern over multiple scales of time or space. The same pattern repeats itself recursively as the scale is further magnified [14]. Self-similarity is also related to other concepts: fractals and chaos theory [11]. The repeating pattern can occur on multiple dimensions. Figure 3.1 shows an example of self-similarity in one dimension. This example may represent a time series viewed over multiple time scales. Figure 3.2 shows an example of self-similarity in two dimensions (referred to as spatial self-similarity). The two examples were constructed using a special mathematical function known as the "Cantor set" [23]. This type of self-similarity where the pattern is completely reproduced on all scales is termed "exact" or "deterministic" self-similarity. In real life phenomena, exact patterns do not hold indefinitely. Rather, the pattern may exhibit some resemblance in the shape over a wide range of scales. This type of self-similarity is referred to as "statistical" or "stochastic" self-similarity. Stochastic self-similarity is observed in many real life phenomena such as natural landscapes, earthquakes, ocean waves and fluctuations in stock market [11]. It has also been observed on traffic flows in communication networks. This chapter

illustrates statistical self-similarity and presents the probability distributions that are used in the queueing analysis performed in this dissertation.

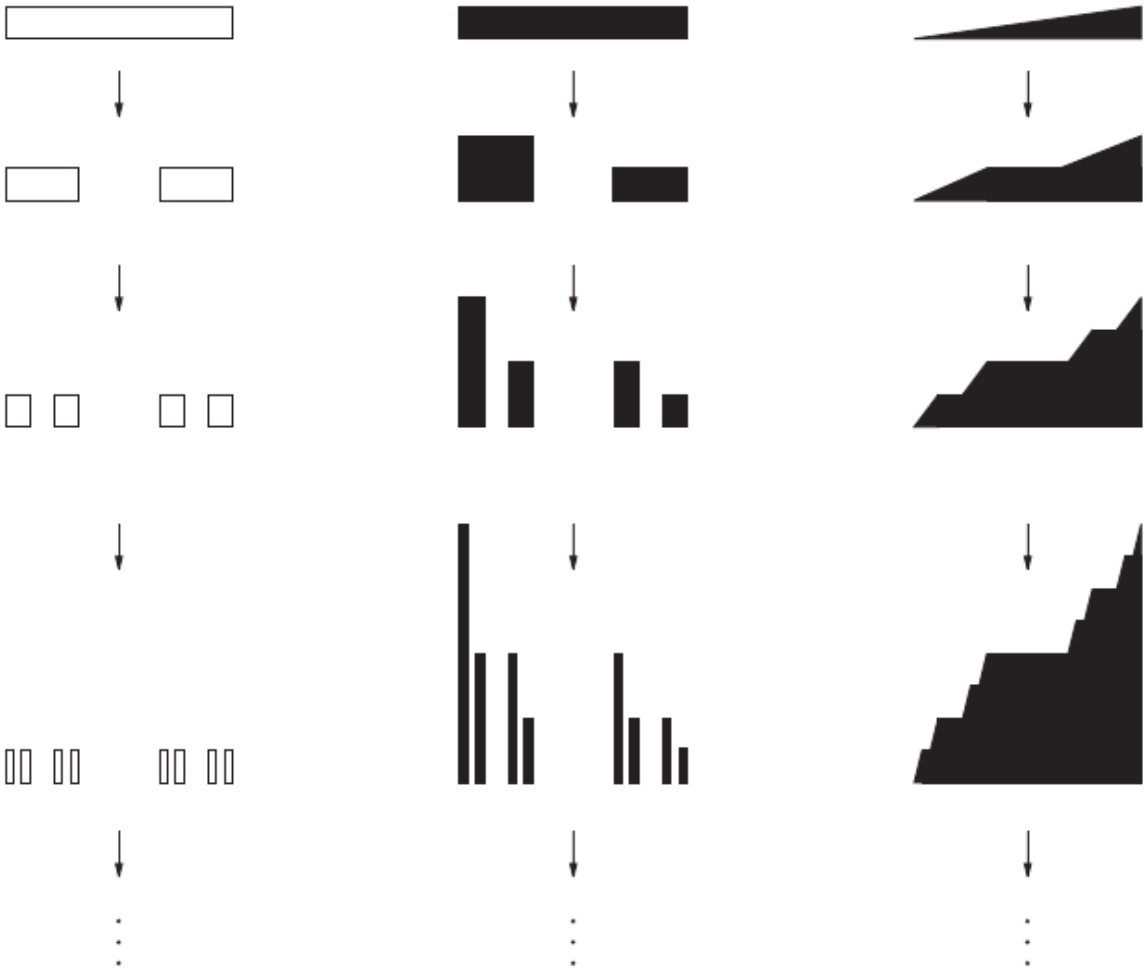

Figure 3.1: Example of self-similarity in a single dimension [14]

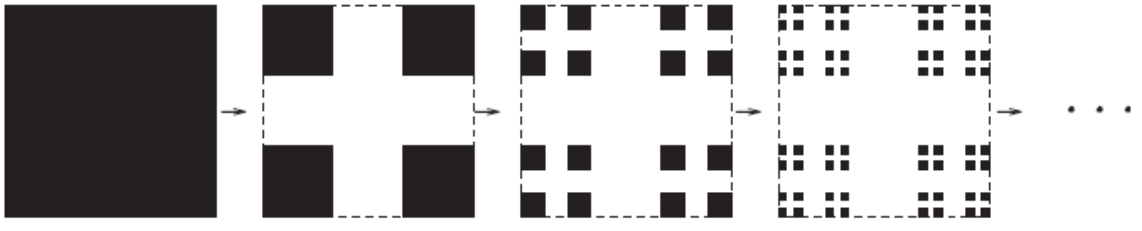

Figure 3.2: Example of self-similarity in two dimensions [14]

3.2 Statistical Self-Similarity

Statistical self-similarity is mathematically characterized by self-similar stochastic processes. Figure 3.3 shows an example of statistical self-similar time series contrasted with a non-self-similar time series.

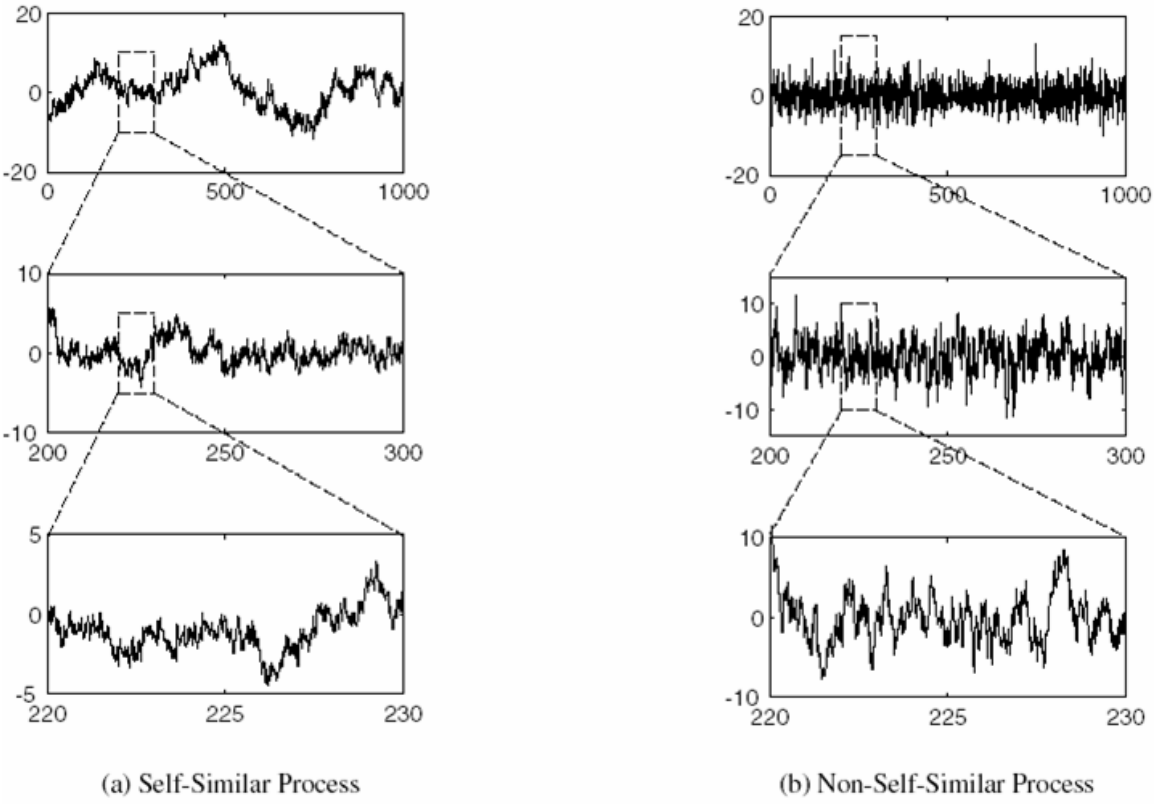

Figure 3.3: Statistical self-similar time series (a) compared to non-self-similar time series (b) [11]

As can be seen in Figure 3.3(a), the time series is not reproduced exactly on magnified time scales. However, the process patterns exhibit resemblance in their shapes [11]. Such resemblance is observed in traffic traces of data networks.

3.3 Self-Similarity in Data Traffic

Self-similarity was first observed on Ethernet Local Area Network (LAN) traffic by a study that was presented in 1993 [15] and later extended in 1994 [24]. The

results of this study were based on measurements of several Ethernet LANs at Bellcore for a period of two and half years. The study used extensive trace data recorded from hundreds of millions of packets with various types of network load conditions. Measurement results of this study are shown in Figure 3.4.

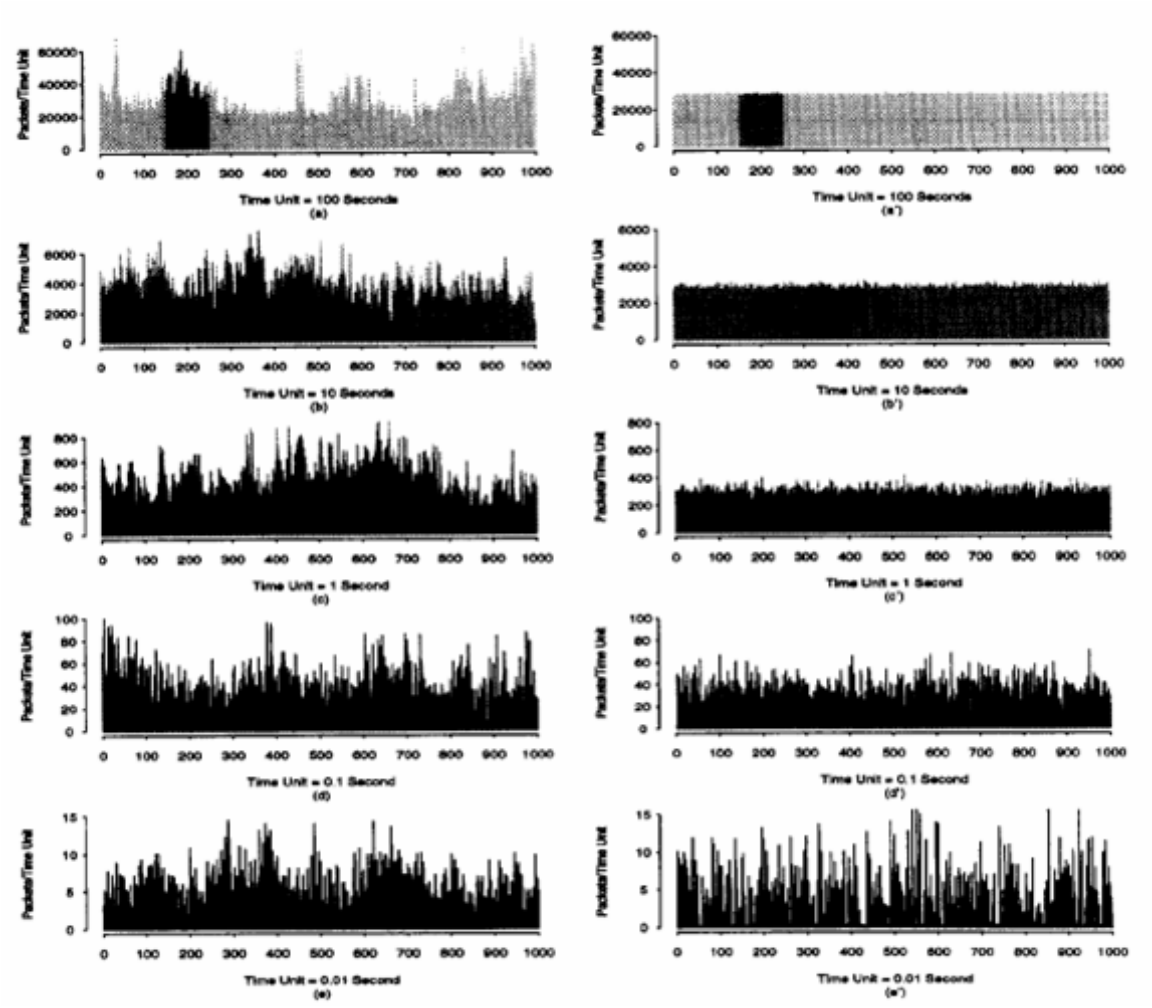

Figure 3.4: Packet arrivals plotted against different time scales on Ethernet LANs which demonstrate the self-similarity of Ethernet traffic [24]

In Figure 3.4 (a)-(e) packet counts per unit time is plotted against multiple time scales. In (a`)-(e`), synthetic Poisson traffic is shown for comparison. Note the similarity between Figure 3.4 and Figure 3.3.

Contrary to the general perception about data networks, a major finding of [24] was to show that variations in small time scales do not fade out in the long run. Consequently, classical Poisson models such as M/M/1 do not accurately describe the actual behavior of data traffic, as shown in Figure 3.4 (a`)-(e`). It wasn't surprising that this study has stimulated numerous other studies on the same subject that confirmed similar results on other types of networks and applications [11]. A study done by Paxon [25] has shown that, in Wide Area Networks, Poisson processes are only valid for modeling sessions initiated by the user for some TCP applications such as TELNET and FTP, but they are invalid for other network processes. Another study [26] has shown that the World Wide Web (WWW) traffic is self-similar. By collecting log information from multiple web servers nationwide, this study has suggested that self-similarity in the World Wide Web is caused by the heavy-tailed distribution of file sizes on the web [26].

3.4 Analytical Modeling

Despite the attractive properties of Poisson based models, which made it possible to develop simple mathematical treatment for M/M/1 queues, these models are no longer adequate for network resource dimensioning due to the self-similar nature of traffic. Actual measurements have shown that classical queueing analysis resulted in rather optimistic performance results that did not match actual traces [11]. As such, it was imperative to consider newer mathematical models that are more appropriate for self-similar traffic. As discussed in Chapter 2, M/G/1 and G/G/1 queue models can be used to characterize traffic with general distribution of service time and interarrival times, respectively. In order to utilize these models, it is first

required to describe the self-similar stochastic process using appropriate probability distribution functions.

This section provides the mathematical definition of self-similar processes and illustrates the concepts of long-range dependence and heavy-tailed distributions, which are tightly related to self-similarity. The probability distributions presented in this section are used in subsequent chapters for delay calculation in the M/G/1 and G/G/1 queue models.

3.4.1 Mathematical Description

A stochastic process is $X(t)$ considered as self-similar process if it satisfy the following conditions [11]:

$$E\left[X\left(t\right)\right] = \frac{E\left[X\left(at\right)\right]}{a^{H}} \tag{3.1}$$

$$\text{var}\left[X\left(t\right)\right] = \frac{\text{var}\left[X\left(at\right)\right]}{a^{2H}} \tag{3.2}$$

$$R_{x}\left(t,s\right) = \frac{R_{x}\left(at,as\right)}{a^{2H}} \tag{3.3}$$

where $a > 0$ is any real number, H is the Hurst parameter and R_x is the autocorrelation function. The Hurst parameter is commonly used to measure the degree of self-similarity or long range dependence. Equations (3.1), (3.2) and (3.3) state that the two processes $a^{-H}X\left(at\right)$ and $X(t)$ have the same statistical properties, which what would be expected as invariance against different time scales.

3.4.2 Long-Range Dependence

A stochastic process is considered as a long-range dependent process if its autocorrelation function decays hyperbolically, which implies that is not summable. Not all self-similar processes are long-range dependent. Similarly, not all long-range processes are self-similar. However, in data traffic, the Hurst parameter range is $0.5 \le H \le 1$. Within this range, self-similarity and long range dependence imply each other and the two terms are used interchangeably [14]. The degree of long range dependence can be expressed by the value of Hurst parameter. The closer the value of H is to 1, the higher the degree of correlation. H=0.5 indicates that there is no correlation or long range dependence[11].

3.4.3 Heavy-Tailed Distributions

Heavy-tailed distributions are defined as follows: a random variable Z is heavy-tailed if:

$$\Pr[Z > x] \sim cx^{-\alpha}, \qquad x \to \infty \tag{3.4}$$

where $0 < \alpha < 2$. This indicates that the tail of the distribution function asymptotically decays hyperbolically. This is in contrast to light-tailed distributions, such as the exponential distribution, where the tail asymptotically decays exponentially. This effectively means that heavy-tailed distributions can take very high values with non-negligible probability [14]. The parameter α is called the shape parameter. Long range distributions have an infinite variance for $0 < \alpha \le 1$. For $1 < \alpha < 2$, both mean and variance are infinite.

Heavy-tailed distributions are tightly related to long-range dependence. Empirical studies have demonstrated a close match between heavy-tailed distributions, such as the Pareto distribution and lognormal distribution, and actual traffic traces of many internet applications [27]. These two distributions are discussed in the remaining part of this section.

The Pareto distribution [28] is the simplest heavy-tailed distribution. A random variable X is said to have Pareto distribution if its probability distribution function (PDF) is described as follows:

$$f(x) = 1 - \left(\frac{\beta}{x}\right)^{\alpha}, \qquad x \geq \beta \tag{3.5}$$

The parameter β is called the location parameter, which is the minimum value the random variable can take. The parameter α is the shape parameter, which determines the mean and variance of the random variable [11, 14].

The mean and variance of the Pareto distribution are calculated as follows:

$$E[X] = \frac{\alpha\beta}{\alpha - 1} \tag{3.6}$$

$$\text{var}[x] = \frac{\alpha\beta^2}{(\alpha - 1)^2(\alpha - 2)} \tag{3.7}$$

Another interesting heavy-tailed distribution is the lognormal distribution. The lognormal distribution is a random variable whose logarithm follows a normal distribution. It has the following distribution function [29]:

$$f(x) = \frac{1}{\alpha\sqrt{2\pi}x} e^{-(\ln x - \beta)^2 / (2\alpha^2)} \tag{3.8}$$

The mean and variance of the lognormal distribution are given by [29]:

$$E[X] = e^{\frac{\alpha^2}{2}+\beta} \tag{3.9}$$

$$\text{var}[X] = e^{\alpha^2+2\beta}\left(e^{\alpha^2}-1\right) \tag{3.10}$$

The lognormal and Pareto distributions are used in both analytical and simulation tests for M/G/1 and G/G/1 queues in subsequent chapters.

3.5 Summary

This chapter has presented the phenomenon of self-similarity and how it applies to network traffic. A literature review of the self-similar nature of network traffic has been provided. Two other concepts related to self-similarity have been illustrated: long range dependence and heavy-tailed distributions. Finally, mathematical models for self-similar traffic characterization have been presented. These models are used in subsequent chapters for analytical and simulation studies.

Chapter 4. Segregation and Integration

Abstract: In this chapter, segregation and integration are studied as two fundamental techniques for queue and bandwidth management. Previous work on the tradeoff between the two techniques using classic queuing models and limited number of flows is reviewed. The delay analysis is extended to M/G/1 and G/G/1 queues with an arbitrary number of flows. The contents of this chapter have partially been published in [30, 31].

4.1 Segregation versus Integration

Segregation and integration are two different techniques for sharing network resources among flows. The segregated system is the one in which the each flow is assigned a separate queue and transmission channel. Queue segregation is achieved by allocating a certain amount of buffer to each flow. Channel segregation can be performed logically by reserving a certain amount of bandwidth for each flow or physically by assigning a separate link (e.g. wavelength in optical networks) to each flow, depending on the underlying physical medium. The integrated system completely shares the buffer and transmission channel among all flows. In this chapter, the two techniques are evaluated in terms of the average packet delay as the disparity between the different flows is changed. It is assumed that packets are served on a First-In-First-Out (FIFO) basis. A schematic comparison between the integrated and segregated approaches is shown in Figure 4.1.

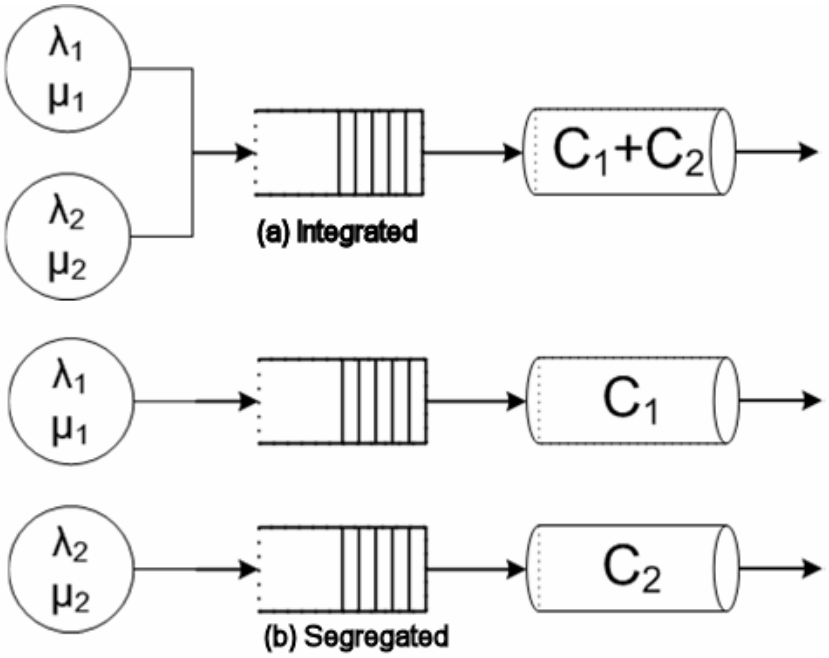

Figure 4.1: Schematic view of the integrated and segregated systems [19]

4.2 Related Work

Theoretical studies of classical queueing models have previously shown that integrating traffic flows with disparate characteristics results in a higher latency than segregating them into fixed bandwidth channels [19, 32]. Comparable results were also reported by more recent studies on networks and network applications [33-36]. In [19], the analysis was based on the M/M/1 queueing model. In [32], the geometric distribution was used for modeling the service times. In this section, the previous analytical work is reviewed. The following sections generalize the previous results and examine how they can be applied to M/G/1 and G/G/1 queueing models.

4.2.1 Delay Analysis for Two Flows with M/M/1 Queue

Rudin [19] has studied the delay of the integrated and segregated systems for two flows of traffic with exponential interarrival and exponential service times (M/M/1 model).

For the segregated system with two flows, the average delay per packet is calculated as follows:

$$T_{seg} = \frac{\lambda_1}{\lambda_1 + \lambda_2} T_1 + \frac{\lambda_2}{\lambda_1 + \lambda_2} T_2 \tag{4.1}$$

where λ_1 and λ_2 are the average arrival rates for flow 1 and flow 2, respectively. T_1 and T_2 are the average delays for flow 1 and flow 2, respectively. Note that T_1 and T_2 can be calculated from (2.8) as follows:

$$T_1 = \frac{1}{\mu_1 C_1 - \lambda_1} \tag{4.2}$$

$$T_2 = \frac{1}{\mu_2 C_2 - \lambda_2} \tag{4.3}$$

where C_1 and C_2 are the channel capacities allocated to flow 1 and flow 2, respectively. Thus:

$$C = C_1 + C_2 \tag{4.4}$$

where C is the total available capacity. The allocation of channel capacity is discussed in section 4.2.3.

The delay for the integrated system with two flows can be calculated using P–K formula, as follows:

$$T_{\text{int}} = \frac{\rho^2 + \lambda^2 \sigma_s^2}{2\lambda(1-\rho)} + \frac{1}{\mu C} \tag{4.5}$$

where $\bar{s}$ is the first moment (mean) and σ_s^2 is the second moment (variance) of the service time. Note that in the integrated system, the parameters used to determine the delay are results of combination of two flows. They are calculated as follows:

$$\lambda = \lambda_1 + \lambda_2 \tag{4.6}$$

$$\frac{1}{\mu} = \frac{\lambda_1}{\lambda_1 + \lambda_2}\frac{1}{\mu_1} + \frac{\lambda_2}{\lambda_1 + \lambda_2}\frac{1}{\mu_2} \tag{4.7}$$

$$\bar{s} = \frac{1}{\mu C} = \frac{1}{\lambda C}\left(\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}\right) \tag{4.8}$$

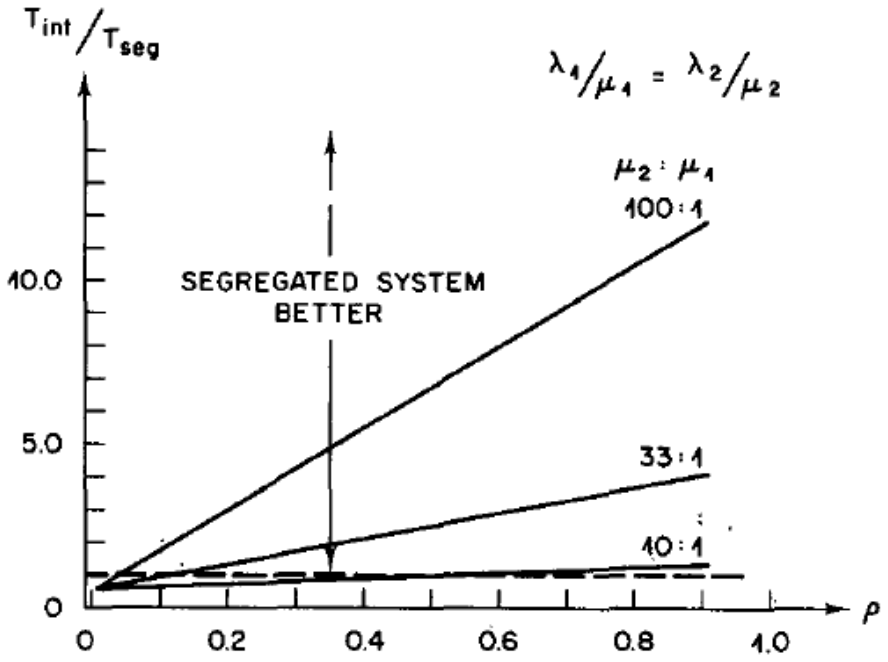

Figure 4.2: T_{int}/T_{seg} as a function of ρ for different ratios of μ_2:μ_1 [19]

The main result obtained by Rudin is summarized in Figure 4.2. It can be seen from Figure 4.2 that the segregated system performs better as the variation of μ_2:μ_1 increases. Rudin explains this behavior as follows. Since the delay calculated is the average delay per packet, a longer packet from one flow will delay multiple short

packets from the other flow. The net result is an increase of the combined average delay per packet [19]. This is a key result that is used extensively in this dissertation.

4.2.2 Delay Analysis for Two Flows with Geometric Service Times

Verma [32] has extended the results obtained by Rudin [19] to queues with geometric service times. This required using different methods for delay calculation and channel capacity allocation. Channel capacity allocation is discussed in more detail in section 4.2.3. The main result of [32] is shown in Figure 4.3.

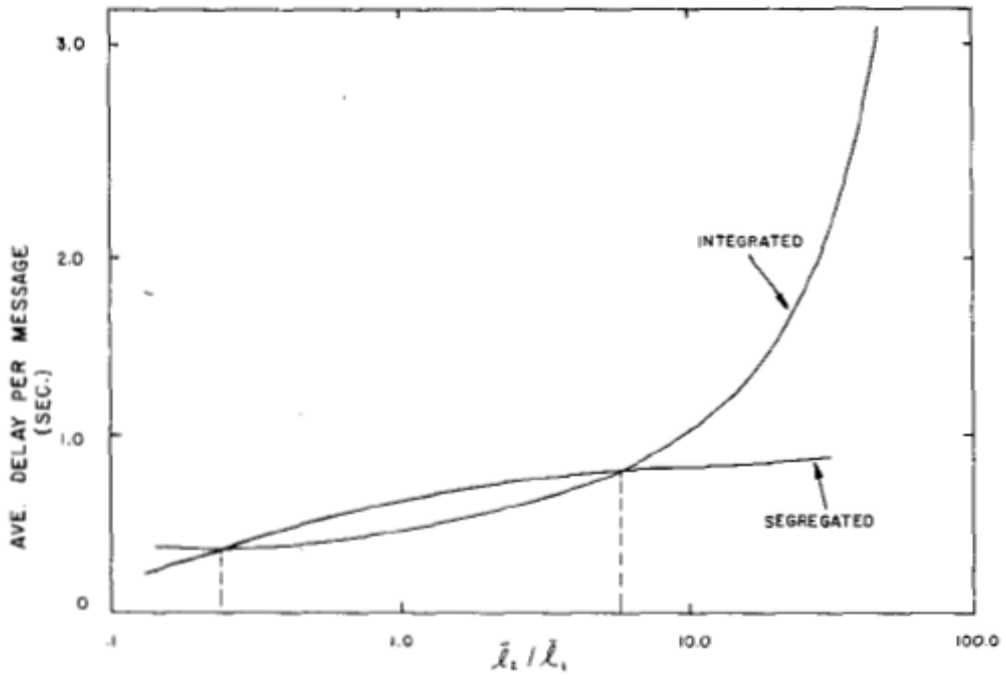

Figure 4.3: Average delay per message for the segregated and integrated system with different ratios of message lengths [32]

From Figure 4.3, it can be seen, in addition to the result obtained in [19], that at some range of the ratio of μ_1:μ_2 (which is the reciprocal of $\overline{l}_2/\overline{l}_1$) the integrated system performs better. Obviously, this range is the range where the variation or the

disparity between packet sizes is small. What is not so obvious is at what point the integrated curve and the segregated curve intersect.

The trade off between integration and segregation was one motivation for developing a system that can combine the advantages of both systems, which is presented in the next chapter. The other motivation was the effect of self-similarity and long range dependence on the average delay. Before discussing the new results, it is worthwhile discussing the channel capacity allocation problem.

4.2.3 Channel Capacity Allocation

In order to make a fair comparison between the integrated and segregated systems, the capacity allocated for each channel in the segregated system must be allocated optimally [19]. This means that the capacity should be allocated in such a way that the delay of the segregated system is at its minimum possible value, with all other parameters being equal. Without this optimal assignment, the comparison might be unfair in favor of the integrated system.

Kleinrock has developed an optimum channel assignment method using the method of Lagrange multipliers [18]. With this method, the value C_i of channel capacity for channel i is calculated as follows:

$$C_i = \frac{\lambda_i}{\mu_i} + C\left(1-\rho\right)\frac{\sqrt{\lambda_i/\mu_i}}{\sum_{j=1}^{n}\sqrt{\lambda_j/\mu_j}} \tag{4.9}$$

where

$$\rho = \frac{\lambda}{\mu C} \tag{4.10}$$

$$\lambda = \sum_{i=1}^{n}\lambda_i \tag{4.11}$$

$$\frac{1}{\mu} = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda} \frac{1}{\mu_i} \tag{4.12}$$

Equation (4.9) can be explained as follows. First, each channel is allocated enough bandwidth to accommodate its flow of λ_i / μ_i . The remaining extra capacity $C - \sum_{i=1}^{n} \sqrt{\lambda_i / \mu_i}$ is distributed on the channels proportionally with the square roots of their mean flows [18].

Note that (4.9) is only applicable to channels with exponential interarrival and service times (M/M/1 queue model). In [19], (4.9) was used in the analysis to calculate the optimum capacity assignment. Although the technique used to derive the optimum value using Lagrange multipliers is applicable for other models, solving the equation for queue models other than M/M/1 usually results in a non-closed form solution. For instance, in [32], differentiating the delay equation with respect to C_1 yielded a sixth order equation that could only be solved numerically.

4.3 Delay Analysis for *n* Flows with M/G/1 Queues

As mentioned in Chapter 3, classical queueing models are considered inadequate for capturing the self-similarity and long range dependence characteristics. In order to utilize the segregation versus integration results obtained in the previous work, it is essential to use analytical models based on heavy tailed distributions of packet arrivals and/or service times are needed. In this section, the analytical framework for the M/G/1 queue with lognormal distribution as a service time distribution is developed. The G/G/1 distribution is studied next.

This section provides the equations used for calculating the delay for both integrated and segregated systems with M/G/1 queue model having heavy-tailed

service time distribution [20]. An important advantage of the M/G/1 queue compared to the G/G/1 queue is the availability of an exact analytical model using the P–K formula [20, 32]. The P–K formula is applicable for any single channel system with exponential interarrival times and any distribution of service times. The average delay per packet for the M/G/1 queue is calculated as follows:

$$T = \frac{\rho^2 + \lambda^2 \sigma_s^2}{2\lambda(1-\rho)} + \frac{1}{\mu C} \tag{4.13}$$

Calculating the delay using (4.13) requires the variance of the service time. As mentioned in subsection 3.4.3, the Pareto distribution has an infinite variance in the range in which it is considered a self-similar process where $1<\alpha<2$. For the lognormal distribution, the variance is finite where $1<\alpha<2$. Thus, the lognormal distribution is used in this analysis for the M/G/1 queue to model the service time. However, the basic derivation is based on the P–K formula and is applicable to any M/G/1 system where the service time has finite first and second moments.

4.3.1 Average Delay for *n* Segregated Flows

The delay for a system of n segregated flows is calculated by generalizing the expression in (4.1):

$$T_{seg} = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda} T_i \tag{4.14}$$

where T_i is the average delay calculated for channel i using (4.13):

$$\begin{aligned} T_i &= \frac{\rho_i^2 + \lambda_i^2 \sigma_{s(i)}^2}{2\lambda_i(1-\rho_i)} + \frac{1}{\mu_i C_i} \\ &= \frac{\lambda_i^2 / \mu_i^2 C_i^2 + \lambda_i^2 \sigma_{s(i)}^2}{2\lambda_i(1-\lambda_i/\mu_i C_i)} + \frac{1}{\mu_i C_i} \end{aligned} \tag{4.15}$$

Let S_i be a random variable representing the service time of flow i. In this case, S_i is a lognormal random variable. From (3.9), the following relations apply:

$$E\left[S_i\right] = \frac{1}{\mu_i C_i} = e^{0.5\alpha_i^2 + \beta_i} \tag{4.16}$$

$$\text{var}\left[S_i\right] = \sigma_{s(i)}^2 = e^{\alpha_i^2 + 2\beta_i}\left(e^{\alpha_i^2} - 1\right) \tag{4.17}$$

Generally, α_i and β_i are specified based on traffic conditions. However, in this analysis, it is required to specify the value of the mean service time in advance. Thus, from (4.16), it can be seen that knowing α_i and the mean service time, then β_i can be calculated as follows:

$$\beta_i = \ln\left(\frac{1}{\mu_i C_i}\right) - \frac{\alpha_i^2}{2} \tag{4.18}$$

Substituting in (4.15) yields:

$$T_i = \frac{\lambda_i^2 \exp\left(\ln\left(\frac{1}{\mu_i C_i}\right) - \frac{\alpha_i^2}{2}\right)\left(e^{\alpha_i^2} - 1\right) + \frac{\lambda_i^2}{\mu_i^2 C_i^2}}{2\lambda_i\left(1 - \lambda_i / \mu_i C_i\right)} + \frac{1}{\mu_i C_i} \tag{4.19}$$

which can be simplified to [31]:

$$T_i = \frac{\lambda_i e^{\alpha_i^2}}{2\mu_i C_i\left(\mu_i C_i - \lambda_i\right)} + \frac{1}{\mu_i C_i} \tag{4.20}$$

4.3.2 Average Delay for *n* Integrated Flows

The average delay for the integrated system T_{int} is calculated using (4.13). To calculate T_{int}, the variance of the service time needs to be calculated. Let G be a random variable representing the service time for the integrated system. The mean service time is calculated by generalizing the expression in (4.8):

$$E[G] = \frac{1}{\mu C} = \frac{1}{\lambda C}\sum_{i=1}^{n}\frac{\lambda_i}{\mu_i} \tag{4.21}$$

The variance of the service time is calculated as follows:

$$E[G] = E\left[G^2\right] - \left(E[G]\right)^2 \tag{4.22}$$

$$\begin{aligned} \mathrm{var}[G] &= E\left[G^2\right] - \left(E[G]\right)^2 \\ &= \sum_{i=1}^{n}\frac{\lambda_i}{\lambda}E\left[S_i^2\right] - \sum_{i=1}^{n}\left(\frac{\lambda_i}{\lambda}E[S_i]\right)^2 \end{aligned} \tag{4.23}$$

Substituting from (4.16) yields [31]:

$$\sigma_s^2 = \mathrm{var}[G] = \sum_{i=1}^{n}\frac{\lambda_i}{\lambda}\exp\left(\alpha_i^2 - 2\ln(\mu C)\right) - \sum_{i=1}^{n}\left(\frac{\lambda_i}{\lambda}\frac{1}{\mu C}\right)^2 \tag{4.24}$$

T_{int} can then be calculated using (4.13) knowing the values of λ_i and μ_i, (i=1,2,…,n). The value of α can be assumed as a constant design parameter.

4.4 Delay Analysis for *n* Flows with G/G/1 Queues

For G/G/1 queues, there is no simple closed form solution for calculating the delay. However, there are methods for calculating an upper bound for the delay. Among the simplest but most powerful approximations is the Kingman's upper bound [21]:

$$T \le \frac{\sigma_\tau^2 + \sigma_s^2}{2\left(\bar{\tau} - \bar{s}\right)} + \bar{s} \tag{4.25}$$

where $\bar{\tau}$ is the mean interarrival time, σ_τ^2 is the variance of the interarrival time, $\bar{s}$ is the mean service time and σ_s^2 is the variance of the service time. Kingman's bound provides a better approximation under high loads as $\rho \to 1$. Other bounds and

approximations are available for the G/G/1 queues [37, 38]. However, the focus in this dissertation is more on the delay comparison, rather than the delay values of the integrated and segregated approaches. Therefore, the Kingman's bound will be used because of its simplicity. As shown in section 5.5 by comparing the analytical and simulation results, the Kingman's bound provides a good approximation to actual delay values.

Note that the variance of the interarrival time and the variance of the service time are both needed to calculate the bound on the delay. Considering that some heavy-tailed distributions have infinite variance, this limits the possible distributions that can be used with this bound. In the following analysis, the lognormal distribution will be used for both interarrival and service times. In the simulation, both lognormal and Pareto distributions will be used. The notation LN/LN/1 queue is used for the lognormal interarrival and lognormal service time queue. The notation P/LN/1 queue is used for the Pareto interarrival and Pareto service time queue. Finally, the notation P/P/1 queue is used for Pareto interarrival and Pareto service time queue.

4.4.1 Average Delay for *n* Segregated Flows

The average delay for the segregated system is calculated using (4.14), where T_i for each channel $i = 1, \cdots, n$ is approximated using (4.25), as follows:

$$T_i \le \frac{\sigma^2_{\tau(i)} + \sigma^2_{s(i)}}{2\left(\overline{\tau}_i - \overline{s}_i\right)} + \overline{s}_i \tag{4.26}$$

which can be rewritten as:

$$T_i \le \frac{\sigma_{\tau(i)}^2 + \sigma_{s(i)}^2}{2\left(\frac{1}{\lambda_i} - \frac{1}{\mu_i C_i}\right)} + \frac{1}{\mu_i C_i} \tag{4.27}$$

To apply (4.27) to LN/LN/1 queues, the following relations apply:

$$\overline{s_i} = \frac{1}{\mu_i C_i} = e^{\frac{\alpha_s^2(i)}{2}} + \beta_{s(i)} \tag{4.28}$$

$$\sigma_{s(i)}^2 = e^{\alpha_s^2(i)+2\beta_s(i)}\left(e^{\alpha_s^2(i)} - 1\right) \tag{4.29}$$

Note the use of *s* qualifier for the lognormal distribution parameters α and β of the service time. This is to distinguish these parameters from the ones used for the interarrival time, which are qualified with τ.

From (4.284.18):

$$\beta_{s(i)} = \ln\left(\frac{1}{\mu_i C_i}\right) - \frac{\alpha_{s(i)}^2}{2} \tag{4.30}$$

Substituting in (4.29) yields:

$$\sigma_{s(i)}^2 = \exp\left(\alpha_{s(i)}^2 + 2\left(\ln\left(\frac{1}{\mu_i C_i}\right) - \frac{\alpha_{s(i)}^2}{2}\right)\right)\left(e^{\alpha_s^2(i)} - 1\right) \tag{4.31}$$

which can be further simplified to:

$$\sigma_{s(i)}^2 = \frac{e^{\alpha_{s(i)}^2} - 1}{\mu_i^2 C_i^2} \tag{4.32}$$

Similarly, for the interarrival time:

$$\sigma_{\tau(i)}^2 = \lambda_i^2\left(e^{\alpha_{\tau(i)}^2} - 1\right) \tag{4.33}$$

Substituting in (4.27) yields:

$$T_i \le \frac{\lambda_i^2\left(e^{\alpha_s^2(i)}-1\right)+\mu_i^2 C_i^2\left(e^{\alpha_\tau^2(i)}-1\right)}{2\lambda_i\mu_i C_i\left(\mu_i C_i-\lambda_i\right)}+\frac{1}{\mu_i C_i} \tag{4.34}$$

4.4.2 Average Delay for *n* Integrated Flows

The average delay for the integrated system is approximated using Kingman's bound (4.25):

$$T_{\text{int}} \le \frac{\sigma_\tau^2+\sigma_s^2}{2\left(\frac{1}{\lambda}-\frac{1}{\mu C}\right)}+\frac{1}{\mu C} \tag{4.35}$$

Applying this bound to the LN/LN/1 queue, the two parameters that need to be calculated are σ_s^2 and σ_τ^2. σ_s^2 is calculated exactly as in section 4.3.2:

$$\sigma_s^2 = \sum_{i=1}^{n}\frac{\lambda_i}{\lambda}\exp\left(\alpha_{s(i)}^2-2\ln\left(\mu C\right)\right)-\sum_{i=1}^{n}\left(\frac{\lambda_i}{\lambda}\frac{1}{\mu C}\right)^2 \tag{4.36}$$

Calculating σ_τ^2 is not as straightforward as calculating σ_s^2 for the integrated system. In the case of service times, the distribution of the combined service time process for the integrated system is calculating by simply adding the service time distributions of the individual Flows factored by their contribution to the packet arrival as in (4.12), (4.21) and (4.23). This is not the case for the interarrival process. Although the distribution of the combined arrival process can be obtained by adding the distributions of the arrival distributions of the individual flows, knowing the distribution of the arrival process does not imply knowing the distribution of the interarrival process. An exception to this case is when the arrival process is Poisson, in which the interarrival distribution is known to be exponential (See Chapter 2). A similar duality doesn't exist for the case where the interarrival distribution is

lognormal. The mean arrival rate of the integrated system can be calculated by simply adding the mean arrival times of the individual flows, regardless of the distribution of the arrival processes of the individual flows. However, the variance cannot be determined without knowing the distribution of the combined arrival process. Even when all flows have the same arrival distribution, the resulting arrival distribution when integrating them into one channel isn't necessarily the same distribution. Some exceptions exist for some distribution, but this is not the general case.

For the sake of simplicity, in the following analytical results, the same interarrival probability distribution model (lognormal) will be used for all flows. It will be further assumed that the interarrival distribution of the integrated system is the same as for the individual flows. Using this assumption, the interarrival process of the integrated system is a lognormal distribution with mean $\overline{\tau} = 1/\lambda$ where $\lambda = \sum_{i=1}^{n} \lambda_i$. From the properties of the lognormal distribution (3.9), the following relation applies:

$$\overline{\tau} = \frac{1}{\lambda} = e^{\frac{\alpha_\tau^2}{2} + \beta_\tau} \tag{4.37}$$

The variance of the interarrival time can then be calculated as follows:

$$\sigma_\tau^2 = e^{\alpha_\tau^2 + 2\beta_\tau} \left(e^{\alpha_\tau^2} - 1 \right) \tag{4.38}$$

From (4.37):

$$\beta_\tau = \ln\left(\frac{1}{\lambda}\right) - \frac{\alpha_\tau^2}{2} \tag{4.39}$$

Substituting in (4.38):

$$\sigma_\tau^2 = \exp\left(\alpha_\tau^2 + 2\left(\ln\left(\frac{1}{\lambda}\right) - \frac{\alpha_\tau^2}{2}\right)\right)\left(e^{\alpha_\tau^2} - 1\right) \tag{4.40}$$

which can be simplified to:

$$\sigma_\tau^2 = \frac{\left(e^{\alpha_\tau^2} - 1\right)}{\lambda^2} \tag{4.41}$$

The average delay for the integrated system for the LN/LN/1 queue can then be approximated by:

$$T_{\text{int}} \le \frac{\mu C\left(e^{\alpha_\tau^2} - 1 + \lambda^2\sigma_s^2\right)}{2\lambda\left(\mu C - \lambda\right)} + \frac{1}{\mu C} \tag{4.42}$$

4.5 Optimal Channel allocation

The performance advantage of segregation as compared to integration is a key notion in this dissertation. As mentioned in section 4.2.3, optimal channel allocation is required for the comparison to be fair. The channel allocation method developed by Kleinrock [18] is only applicable to M/M/1 queues. Thus, it is not very useful for the M/G/1 and G/G/1 queues which are more appropriate for modeling self-similar traffic. For general queueing types, there is no closed form solution for optimal channel allocations. Pollet [39] has developed an approximation based on a perturbation technique to calculate the optimal channel allocation in general queueing networks. This method requires solving high order equations for practical applications such as communication networks.

To simplify the analysis, but without sacrificing its validity, the total available channel capacity is distributed among each of the segregated channels so that the

utilization ρ for each channel is the same, i.e., $\rho=\lambda_i/(\mu_i C_i)$ for all channels. Thus, the capacity allocated for each channel would be:

$$C_i = \frac{1}{\rho}\frac{\lambda_i}{\mu_i} \tag{4.43}$$

Note that if the channel capacities are assigned in an optimal manner, the advantage of the segregated system will be higher; in other words, the segregated advantage in this analysis presents a pessimistic result.

4.6 Conclusions

This chapter has studied two fundamental resource management techniques: segregation and integration. Previous studies have shown that the integration of flows with disparate packet sizes results in higher average delay per packet for the combined traffic. Those studies where performed on a limited number of flows using classical queueing models. In order to utilize these results in the context of this dissertation, this chapter has provided the analytical model required for studying the delay performance of the segregated and integrated systems with M/G/1 and G/G/1 queues and arbitrary number of flows. The formulas developed in this chapter are used in the next chapter to produce numerical figures and to derive the equations needed to perform the delay analysis for the newly proposed hybrid integration approach.

Chapter 5. The Hybrid Integration Approach

Abstract: This chapter presents the hybrid integration approach, which is a significant contribution of this dissertation. The first part of the chapter describes the concept of the new approach. Next, the analytical framework for delay calculation is discussed in continuation to the analysis presented in the previous chapter. Details of the hybrid grouping allocation are then presented. Finally, the performance advantage of this approach is demonstrated by analytical and simulation results. The work of this chapter has been published in [30, 31].

5.1 The Concept of Hybrid Integration

The general idea in the newly proposed hybrid integration approach is grouping flows with comparable characteristics into separate channels and assigning separate queue and channel capacity to each group. The hybrid system is a segregated system on the group level rather than on the flow level. Each group can be considered as a mini integrated channel.

The main feature of this approach is that it is dynamic. From the results obtained in [19, 32] and the results obtained in this chapter, it can be seen that neither the integrated nor the segregated approach provide the lowest delay over the entire range of disparities between average packet sizes. Thus, it is inefficient to design the system as one or the other. The designer has to make the tradeoff decision once, which might not result in the optimal performance under all conditions. With the hybrid integration approach, the system periodically collects information about the

flowing traffic and adjusts groupings accordingly. An example of the hybrid integration approach is shown in Figure 5.1.

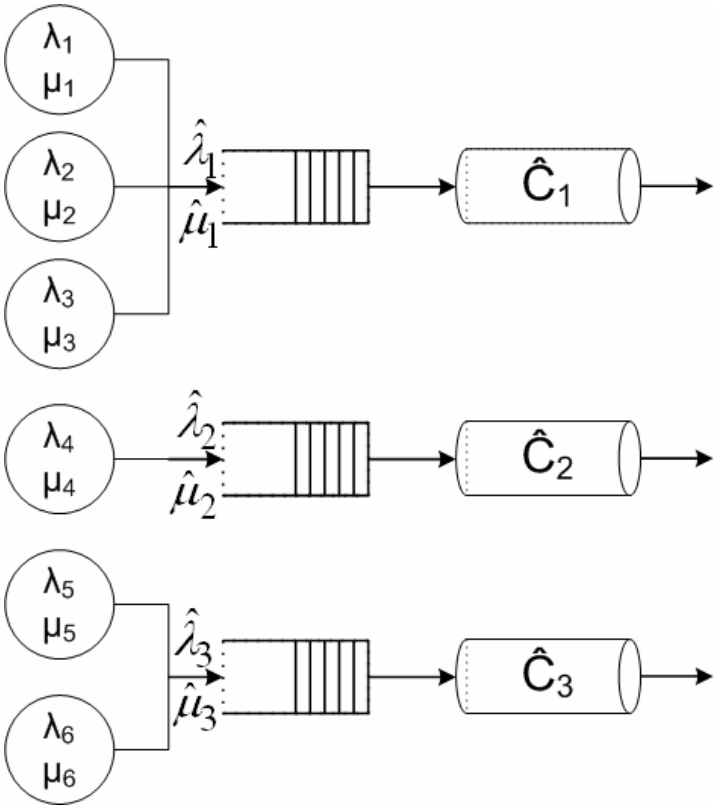

Figure 5.1: Example of a hybrid integration approach with grouping $\{\{1,2,3\},\{4\},\{5,6\}\}$

It is important to note that the pure segregation or integration can be considered as two special cases of the hybrid integration approach. The hybrid system is equivalent to the segregated system when the number of groups equals the number of flows and is equivalent to the integrated system when the number of groups equals one. Thus, the advantage of either approach is not sacrificed.

In designing a switching device with a hybrid integration approach, the most important question to answer is: how to determine the grouping the results in the lowest delay? In order to answer this question, it is first required to establish the analytical framework for calculating the delay in a similar fashion to the analysis

performed in Chapter 4. Next, different methods are discussed for reducing the number of calculations required to determine the optimal grouping.

5.2 Delay Analysis

The hybrid system is a segregated system on the group level. Each group is considered a mini integrated system. To calculate the average delay for the hybrid system, the delay for each group is calculated the same way the delay for the integrated system is calculated, except for the channel capacity, which should be the capacity assigned to this particular group, rather than the total channel capacity. After calculating the delay for each group, the combined average delay for the hybrid system is calculated the same way the delay of the segregated system is calculated. Table 5.1 summarizes the main notations used in the analysis of the hybrid integration approach.

Notation	**Meaning**
K	Number of groups in the system. $1<K<n$
$i \in j$	all flows i that belong the group j
$\hat{C}_j$	Channel capacity assigned to group j
$\hat{\lambda}_j$	Mean arrival rate for group j
$1/\hat{\mu}_j$	Average packet size for group j
$\hat{\rho}_j$	Utilization of the channel used by group j. $\hat{\rho}_j = \hat{\lambda}_j / \hat{\mu}_j \hat{C}_j$
$\hat{\bar{s}}_j$	Mean service time for group j
$\hat{\sigma}^2_{\tau j}$	Variance of the interarrival time for group j
$\hat{\sigma}^2_{sj}$	Variance of the service time for group j
$\hat{T}_j$	Average delay per packet for group j
T_{hyb}	Average delay per packet for the hybrid system

Table 5.1: Notations used in the hybrid integration approach

The following relations are defined based on the notations in Table 5.1:

$$\hat{\lambda}_j \triangleq \sum_{i \in j} \lambda_i \tag{5.1}$$

$$\hat{C}_j \triangleq \sum_{i \in j} C_i \tag{5.2}$$

The delay for the hybrid system can be calculated as follows:

$$T_{hyb} = \sum_{j=1}^{K} \hat{T}_j \frac{\hat{\lambda}_j}{\lambda} \tag{5.3}$$

$\hat{T}_j$ for each group *j* can be calculated using the equations provided in Chapter 4, namely (4.13) for M/G/1 queues or (4.42) for G/G/1 queues. Thus, for the M/G/1 queue:

$$\hat{T}_j = \frac{\hat{\rho}_j^2 + \hat{\lambda}_j^2 \hat{\sigma}_{sj}^2}{2\hat{\lambda}_j \left(1 - \hat{\rho}_j\right)} + \frac{1}{\hat{\mu}_j \hat{C}_j} \tag{5.4}$$

For the G/G/1 queue:

$$\hat{T}_j \leq \frac{\hat{\sigma}_{\tau j}^2 + \hat{\sigma}_{sj}^2}{2\left(1/\hat{\lambda}_j - 1/\hat{\mu}_j \hat{C}_j\right)} + \frac{1}{\hat{\mu}_j \hat{C}_j} \tag{5.5}$$

For the G/G/1 queue with lognormal interarrival and service times:

$$\hat{T}_j \leq \frac{\hat{\mu}_j \hat{C}_j \left(e^{\alpha_\tau^2(j)} - 1 + \hat{\lambda}_j^2 \hat{\sigma}_{sj}^2\right)}{2\hat{\lambda}_j \left(\hat{\mu}_j \hat{C}_j - \hat{\lambda}_j\right)} + \frac{1}{\hat{\mu}_j \hat{C}_j} \tag{5.6}$$

5.3 Grouping Analysis

One problem in designing the hybrid integrated system is fact that there are several ways in which flow grouping can be done. Each way will simply be termed as "grouping". Different groupings result in different average delays. To ensure that the hybrid integration approach results in the lowest possible delay, in general, all

possible groupings need to be examined against the delay resulting from each grouping. This also requires the availability of an analytical model for calculating T_{hyb}.

5.3.1 All Possible Groupings

A system with n flows can be arranged into a minimum of one group (one possible grouping) and a maximum of n groups (one possible grouping). Between these two special cases, there are different ways in which groupings can be made. Let A_{nj} denote the number of groupings for n flows in j groups, where $1 \le j \le n$. R_n = Total number of groupings for n flows. It follows that

$$R_n = \sum_{j=1}^{n} A_{nj} \tag{5.7}$$

As stated above, the following relation applies:

$$A_{(n,1)} = 1 \tag{5.8}$$

The number of groupings of n flows into j groups can be calculated as follows:

$$A_{nj} = A_{(n-1,j-1)} + j \cdot A_{(n-1,j)} \qquad (n > 1, j > 1) \tag{5.9}$$

Equation (5.9) can be explained as follows. Let S_{nj} denote the state in which n flows are arranged in j groups. S_{nj} can be reached either from $S_{n-1,j}$ or $S_{n-1,j-1}$. In the first case, there are already j groups and the n^{th} flow can only be placed in one of the j groups, which have $A_{n-1,j}$ possible groupings, yielding a total of $j\,A_{n-1,j}$ possibilities. In the second case, the n^{th} flow is placed in a new group by itself. The number of

groupings in this case is the same as with $j-1$ groups, i.e., $A_{n-1,j-1}$. Table 5.2 shows the number of groupings for different number of flows.

n	**4**	**8**	**10**	**25**
R_n	15	877	115971	4.68×10^{18}

Table 5.2: Number of flows n and the corresponding total number of groupings R_n

5.3.2 Grouping with Sorted List

As seen from Table 5.2, the number of possible groupings increases rapidly with increasing the number of flows. For the hybrid approach to be feasible, it is important to reduce the number of possibilities to be evaluated in order to reduce the processing overhead associated with this approach.

From Figure 4.3, it can be seen that as the disparity between the packet sizes of the two flows increases, the delay increases monotonically. It can be inferred from this that if the disparity between, say, flow 1 and flow 3 is greater than between flow 1 and flow 2, then integrating 1 and 3 will result in a higher delay than integrating 1 and 2. Consequently, it can be stated that if the flows are numbered in ascending order according to their mean packet length, then the number of comparisons can be reduced by discarding all grouping arrangements such as $\{\{1,3\},\{2\}\}$ or $\{\{1,4\},\{3\}\}$. It is recognized that this proof is intuitive and heuristic. It has not been attempted to provide a formal proof of this statement in this dissertation. Figure 5.2 shows possible groupings for sorted lists of different number of flows. With this sorting, the number of combinations of groupings to be tested for each additional flow i is 2^{i-1}. Therefore, for n flows the number of groupings to be tested is:

$$R_n = \sum_{i=0}^{n-1} 2^i = 2^n - 1 \qquad (5.10)$$

Table 5.3 shows the number of groupings to be tested when sorting is used.

n	**4**	**8**	**10**	**25**
R_n	8	128	512	16777216

Table 5.3: Number of flows and the corresponding number of groupings to be tested R_n

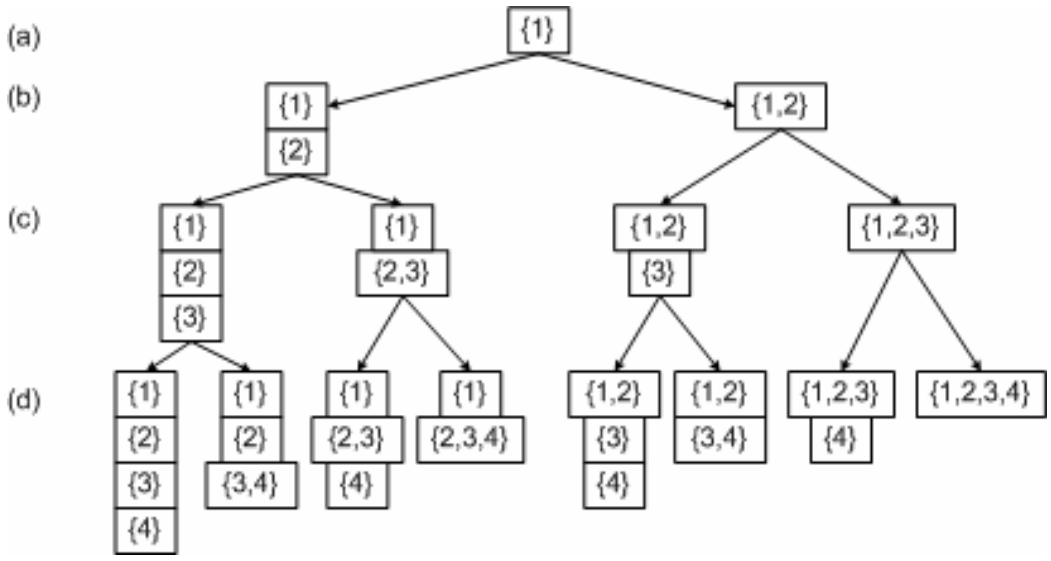

Figure 5.2: Possible groupings for a sorted list of n flows. (a) n=1 (b) n=2 (c) n=3 (d) n=4

5.3.3 Further Optimization

Comparing Table 5.2 and Table 5.3 shows a considerable reduction achieved with sorting. However, the number of calculations is still exponentially related to the number of flows. By analyzing (5.3) and Figure 5.2, a more efficient approach can be obtained for reducing the number of comparisons required. If the grouping $\{\{1,2\}\}$ has a lower delay than $\{\{1\},\{2\}\}$ for instance, then any grouping in lower levels containing $\{\{1\},\{2\}\}$ has a higher delay than its corresponding grouping containing $\{\{1,2\}\}$ and hence need not to be tested. The proof can be demonstrated by the following example:

$$T_{(\{1\},\{2\})} = \frac{\lambda_1}{\lambda_1+\lambda_2}T_1 + \frac{\lambda_2}{\lambda_1+\lambda_2}T_2 \tag{5.11}$$

$$\begin{aligned} T_{(\{1\},\{2\},\{3\})} &= \frac{\lambda_1}{\lambda_1+\lambda_2+\lambda_3}T_1 + \frac{\lambda_2}{\lambda_1+\lambda_2+\lambda_3}T_2 + \frac{\lambda_3}{\lambda_1+\lambda_2+\lambda_3}T_3 \\ &= \left(\frac{\lambda_1}{\lambda_1+\lambda_2}T_1 + \frac{\lambda_2}{\lambda_1+\lambda_2}T_2\right)\frac{\lambda_1+\lambda_2}{\lambda_1+\lambda_2+\lambda_3} + \frac{\lambda_3}{\lambda_1+\lambda_2+\lambda_3}T_3 \\ &= \frac{\lambda_1+\lambda_2}{\lambda_1+\lambda_2+\lambda_3}T_{(\{1\},\{2\})} + \frac{\lambda_3}{\lambda_1+\lambda_2+\lambda_3}T_3 \end{aligned} \tag{5.12}$$

$$T_{(\{1,2\},\{3\})} = \frac{\lambda_1+\lambda_2}{\lambda_1+\lambda_2+\lambda_3}T_{(\{1,2\})} + \frac{\lambda_3}{\lambda_1+\lambda_2+\lambda_3}T_3 \tag{5.13}$$

By comparing (5.12) and (5.13) it can be seen that:

$$T_{(\{1,2\},\{3\})} < T_{(\{1\},\{2\},\{3\})} \leftrightarrow T_{(\{1,2\})} < T_{(\{1\},\{2\})} \tag{5.14}$$

The same observation can be applied with *n* number of flows, as follows:

With *n* flows of traffic, assume the grouping arrangement is as follows:

$\{1,\cdots,a\},\{b,\cdots,c\},\cdots,\{d,\cdots,n\}$, where $1 < a < b < c < n.$ The delay $T_{(n)}$ for this grouping can be calculated as:

$$\begin{aligned} T_{(n)} = {} & \frac{\lambda_1+\cdots+\lambda_a}{\lambda_1+\cdots+\lambda_n}T_{(\{1,\cdots,a\})} + \frac{\lambda_b+\cdots+\lambda_c}{\lambda_1+\cdots+\lambda_n}T_{(\{b,\cdots,c\})} + \cdots \\ & + \frac{\lambda_d+\cdots+\lambda_n}{\lambda_1+\cdots+\lambda_n}T_{(\{d,\cdots,n\})} \end{aligned} \tag{5.15}$$

Suppose an additional flow *n*+1 is to be added in a new group (segregated), the delay $T_{(n+1)}$ can be calculated as:

$$T_{(n+1)} = \frac{\lambda_1 + \cdots + \lambda_a}{\lambda_1 + \cdots + \lambda_{n+1}} T_{(\{1,\cdots,a\})} + \frac{\lambda_b + \cdots + \lambda_c}{\lambda_1 + \cdots + \lambda_{n+1}} T_{(\{b,\cdots,c\})} + \cdots + \frac{\lambda_d + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_{n+1}} T_{(\{d,\cdots,n\})} + \frac{\lambda_{n+1}}{\lambda_1 + \cdots + \lambda_{n+1}} T_{n+1}$$

$$= [\frac{\lambda_1 + \cdots + \lambda_a}{\lambda_1 + \cdots + \lambda_n} T_{(\{1,\cdots,a\})} + \frac{\lambda_b + \cdots + \lambda_c}{\lambda_1 + \cdots + \lambda_n} T_{(\{b,\cdots,c\})} + \cdots \frac{\lambda_d + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_n} T_{(\{d,\cdots,n\})}] \frac{\lambda_1 + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_{n+1}} + \frac{\lambda_{n+1}}{\lambda_1 + \cdots + \lambda_{n+1}} T_{n+1}$$

yielding:

$$T_{(n+1)} = \frac{\lambda_1 + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_{n+1}} T_{(n)} + \frac{\lambda_{n+1}}{\lambda_1 + \cdots + \lambda_{n+1}} T_{n+1} \qquad (5.16)$$

Assume that, before the addition of flow n+1, the n flows could have been arranged in two different groupings A and B. Let $T_{(n)A}$ and $T_{(n)B}$ denote the delay for grouping A and grouping B, respectively. Also, let $T_{(n+1)A}$ and $T_{(n+1)B}$ denote the delay after adding flow n+1 for grouping A and grouping B, respectively. Using (5.16), the delay after adding the n+1 flow is calculated as one of the following:

$$T_{(n+1)A} = \frac{\lambda_1 + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_{n+1}} T_{(n)A} + \frac{\lambda_{n+1}}{\lambda_1 + \cdots + \lambda_{n+1}} T_{n+1} \qquad (5.17)$$

$$T_{(n+1)B} = \frac{\lambda_1 + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_{n+1}} T_{(n)B} + \frac{\lambda_{n+1}}{\lambda_1 + \cdots + \lambda_{n+1}} T_{n+1} \qquad (5.18)$$

By comparing (5.17) and (5.18), it can be seen that:

$$T_{(n+1)A} > T_{(n+1)B} \leftrightarrow T_{(n)A} > T_{(n)B} \qquad (5.19)$$

Using this observation, the optimal grouping algorithm can be stated. The notations used are: i for the tree level and H_i for the array with groupings generated in level i.

1. Start with $i \leftarrow 1$. $H_i \leftarrow \{\{1\}\}$
2. Calculate T_{hyb} for groupings in H_i
3. $L \leftarrow$ minimum T_{hyb} grouping in H_i
4. If $i = n$ then return L and stop
5. Generate the left child (segregated) and right child (integrated) of L. For all other groupings, generate only the right child.
6. Add generated groupings to H_{i+1}
7. $i \leftarrow i+1$. Go to step 2.

Using this algorithm, combinations of groupings to be tested for each additional flow i is equal to i. Therefore, for n flows the number of groupings to be tested is:

$$R_n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{5.20}$$

Table 5.4 shows the number of comparisons needed with the optimized approach. A significant calculation reduction using the optimized approach can be observed by comparing Table 5.4 to Table 5.2 and Table 5.3.

n	**4**	**8**	**10**	**25**
R_n	10	36	55	325

Table 5.4: Number of flows and the corresponding number of groupings to be tested R_n with the optimized approach

5.4 Heuristic Techniques

As already mentioned, the optimal grouping determination techniques discussed in the previous sections require an analytical model for calculating the delay. As mentioned in Chapters 3 and 4, some G/G/1 queue models do not have closed form solutions for calculating the delay. Also, other queues, such as M/G/1 queues with heavy tailed distributions, may have an infinite variance under certain conditions, which makes it impossible to accurately calculate the delay under those conditions, even when the analytical model is available.

In such situations, it is desirable to be able to approximately determine the optimal grouping without a need for calculating the delay. In this part, two heuristic methods are proposed for an approximate determination of the optimal grouping. In both methods, the criterion used to decide whether to combine the two flows or to separate them is the ratio of the average packet sizes between the two flows. If the ratio is larger than a specified threshold (i.e., less difference in average packet size), this means that the disparity between these two flows is not large enough to justify segregation. If the ratio is lower than the threshold, this means that the disparity is high and the two flows are segregated. The two methods differ in the order of which they combine flows.

In the two methods described below, a list of flows 1, …, n is assumed, which is sorted in ascending order according to the average packet size of each flow.

5.4.1 Method 1

In this method, flows are examined in a sequential order. If the next flow is within a close range to the current group, it is added to that group. Otherwise, it is

placed in a new group. This process is examined until the last flow is reached. The algorithm can be stated as follows: Let F_i denote flow number i, $i = 1, \ldots, n$, G_j denote group number j. L_i denote the average packet size of flow i and L_j denote the average packet size of group j.

1. Set $i \leftarrow 1, j \leftarrow 1$
2. Create group G_j (if it doesn't exist)
3. Add flow F_i to group G_j
4. If $i = n$, stop. The current grouping is optimum
5. Set L_j = combined average delay for flows in group j
6. Calculate ratio $R = L_j/L_{i+1}$
7. If $R >$ Threshold, add flow F_{i+1} to group G_j
8. Else, set $j \leftarrow j + 1$
9. Set $i \leftarrow i + 1$
10. Go to step 2

5.4.2 Method 2

From Method 1, it can be observed that the order of which flows are combined can affect the final grouping. Method 2 aims to optimize grouping by combining flows with closest average packet size first. The algorithm can be stated as follows:

1. For $i = 1$ to n, set $j \leftarrow i$ and create group G_j
2. Set k = current number of groups
3. For $j = 1$ to k, calculate L_j
4. For $j = 1$ to k–1, set $R_j = L_j/L_{j+1}$

5. Set $x \leftarrow j$ that corresponds to the maximum R_j ($1 \leq j \leq k$)
6. If R_x < Threshold, stop. The current grouping is optimum
7. Else Combine G_x and G_{x+1}
8. Go to step 2

5.5 Analytical and Simulation Results

This section provides numerical figures for comparing the delay performance of the segregated and integrated approaches with the hybrid integration approach. A similar comparison to the one done in [32] is used in this section. The average delay per packet is evaluated for traffic flows with different disparities.

5.5.1 Input Data

The input data for both analytical and simulation setups were prepared as follows:

- Each flow represents a different source
- The number of flows n=6 flows
- 9 different datasets are used.
- In each dataset, the mean packet sizes and mean arrival rates of the different flows are varied according to some function to increase the disparity in each dataset.
- The utilization factor $\rho=\lambda_i/(\mu_i C_i)$ is kept constant at around 0.9 to make the comparison fair.

Because the number of flows is more than two, the ratio cannot be used to measure the disparity. Instead, the mean packet sizes in the dataset are treated as a

population and the standard deviation of this population is used to measure the disparity. The standard deviation is σ_L calculated as follows:

$$\sigma_L = \sqrt{\sum_{i=1}^{n} \frac{\lambda_i}{\lambda} \left(\frac{1}{\mu_i} - \frac{1}{\mu} \right)^2} \tag{5.21}$$

where $1/\mu$ is calculated from (4.12).

5.5.2 Simulation Setup

Simulations were performed using the Network Simulator ns-2 [34]. UDP packet generators were used for all simulations. The packet generators used random variables of the appropriate types. The channels were represented by simplex links, sources were represented by nodes and traffic flows were represented by UDP agents attached to the corresponding nodes. The integrated system had 1 source node, 1 link and *n* agents. The segregated system had *n* source nodes, *n* links and *n* agents. The hybrid system topology depends on the grouping arrangement for each case. Screenshots of simulation runs demonstrating the simulation setups for the segregated system, the integrated system and the hybrid system are shown in Figure 5.3, Figure 5.4 and Figure 5.5, respectively. The screenshots were taken from the Network Animator tool which is part of the Network Simulator package. The different colors indicate different flows (packet sources). The solid lines represent the actual data being transmitted over the link and the dotted lines show the queues for each flow (packet source).

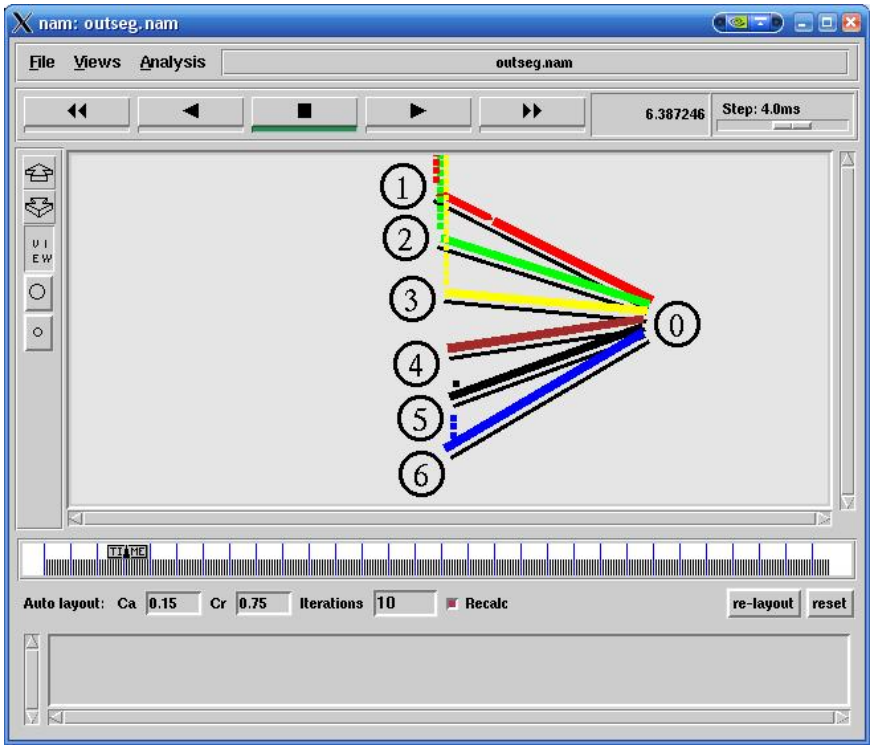

Figure 5.3: Simulation setup for the segregated system

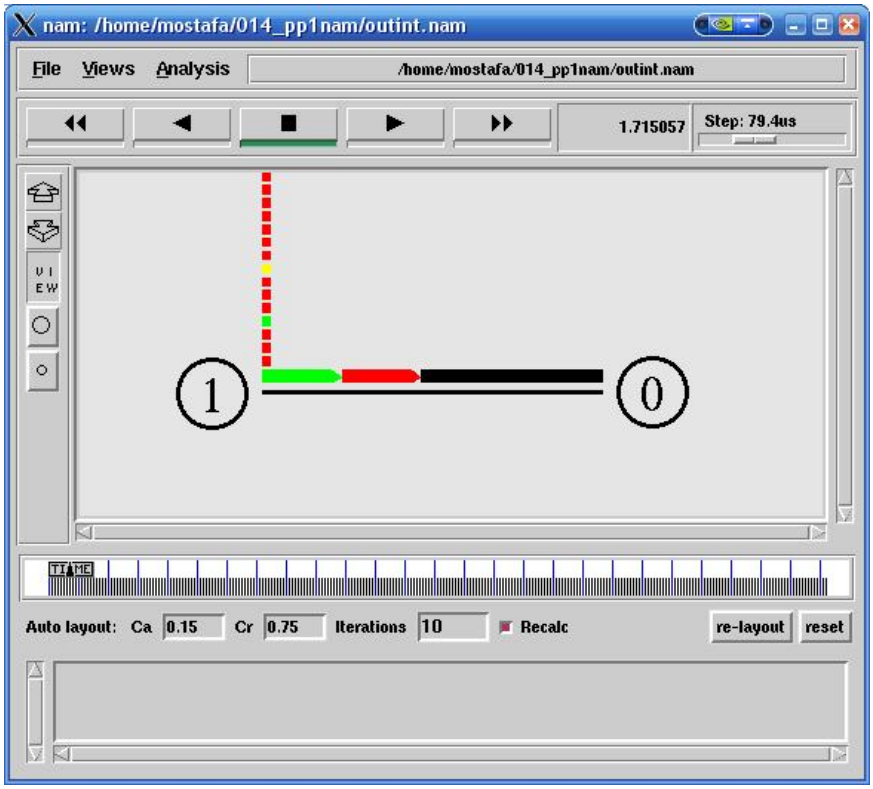

Figure 5.4: Simulation setup for the integrated system

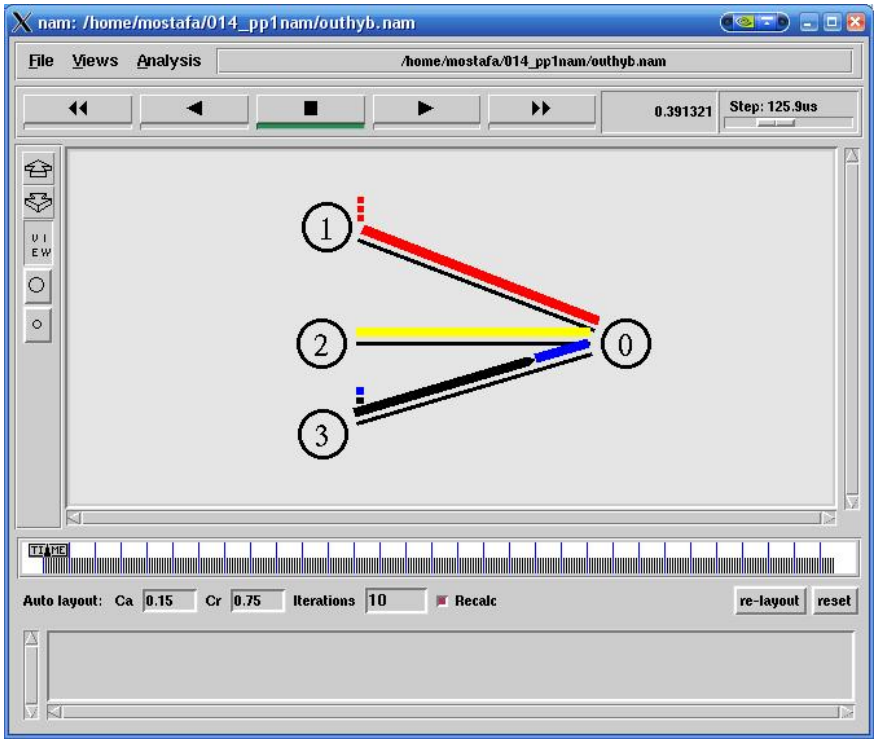

Figure 5.5: Simulation setup for the hybrid integration approach

5.5.3 Results for M/LN/1 queues

For the M/LN/1 queue, a lognormal distribution was used for the service time. The shape parameter value of α=0.5 was used in all datasets. Input data and output results for both analytical and simulation are shown in Table 5.5.

	1	2	3	4	5	6	7	8	9
λ_1	328.5	328.5	328.5	328.5	328.5	328.5	328.5	328.5	328.5
$1/\mu_1$	1	1	1	1	1	1	1	1	1
λ_2	203.4	135.6	101.7	81.36	67.8	58.114	50.85	45.2	40.68
$1/\mu_2$	2	3	4	5	6	7	8	9	10
λ_3	49.725	24.863	14.207	9.945	7.1036	5.3757	4.1438	3.3712	2.7625
$1/\mu_3$	4	8	14	20	28	37	48	59	72
λ_4	8.2286	3.0316	1.44	0.78904	0.48814	0.32542	0.22857	0.16696	0.12632
$1/\mu_4$	7	19	40	73	118	177	252	345	456
λ_5	47.782	11.945	4.4923	2.0857	1.1159	0.65782	0.41582	0.27765	0.19331
$1/\mu_5$	11	44	117	252	471	799	1264	1893	2719
λ_6	23.874	4.536	1.3622	0.53554	0.2495	0.13083	0.07479	0.045666	0.029371
$1/\mu_6$	19	100	333	847	1818	3467	6065	9933	15444
σ_L	4.0533	11.313	21.612	34.986	51.57	71.389	94.526	120.98	150.83
T_{int}	0.02377	0.099115	0.29594	0.70784	1.4577	2.6973	4.6113	7.4132	11.353
T_{int}sim	0.023952	0.10421	0.28378	0.78075	1.7216	2.5391	5.168	7.218	8.5412
T_{seg}	0.055331	0.071983	0.081031	0.086485	0.090318	0.09311	0.095264	0.096943	0.098315
T_{seg}sim	0.055638	0.074282	0.076316	0.091318	0.10035	0.093241	0.10936	0.10835	0.099833
T_{hyb}	0.021521	0.03445	0.048559	0.057357	0.064355	0.070923	0.077595	0.083429	0.087065
T_{hyb}sim	0.022477	0.036266	0.047197	0.061249	0.074649	0.068415	0.082825	0.10196	0.089299
Hybrid Arrgmt	{{1, 2, 3}, {4, 5, 6}}	{{1, 2, 3}, {4, 5, 6}}	{{1, 2, 3}, {4, 5, 6}}	{{1}, {2, 3}, {4, 5, 6}}	{{1}, {2, 3}, {4, 5, 6}}	{{1}, {2, 3}, {4, 5, 6}}	{{1}, {2, 3}, {4, 5, 6}}	{{1}, {2, 3}, {4, 5}, {6}}	{{1}, {2}, {3}, {4, 5}, {6}}

Table 5.5: Analytical and simulation results for M/G/1 queue with lognormal service times

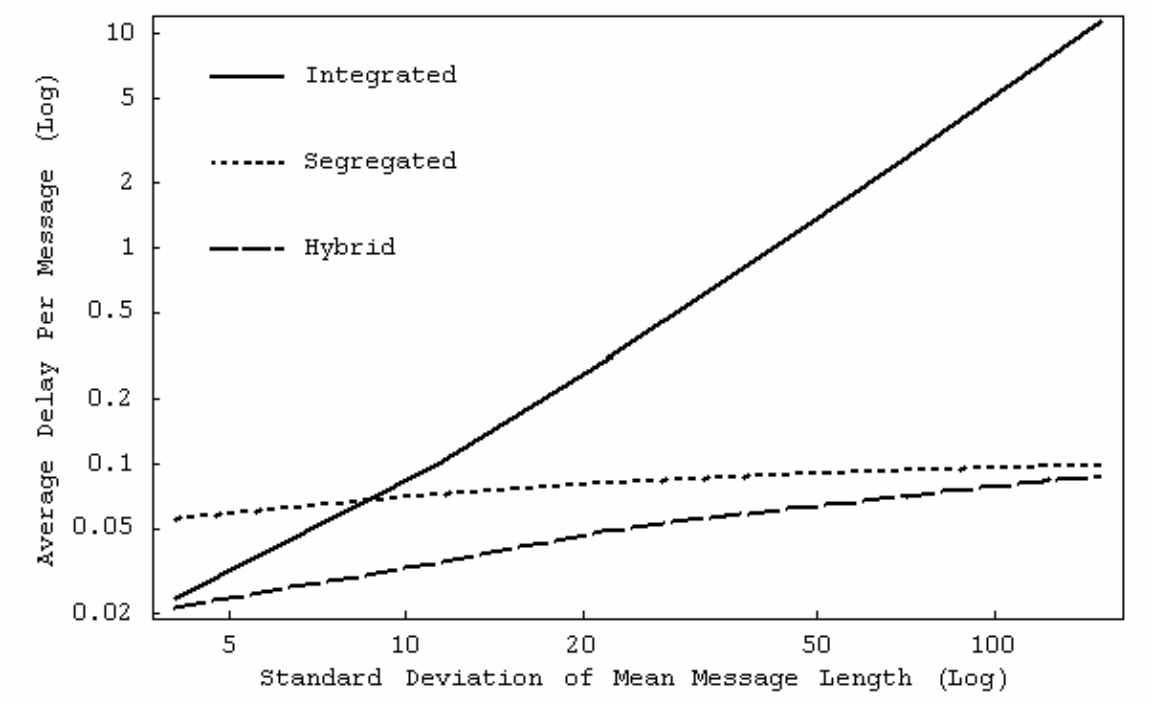

Figure 5.6: Plot of the analytical results for the M/LN/1 queue

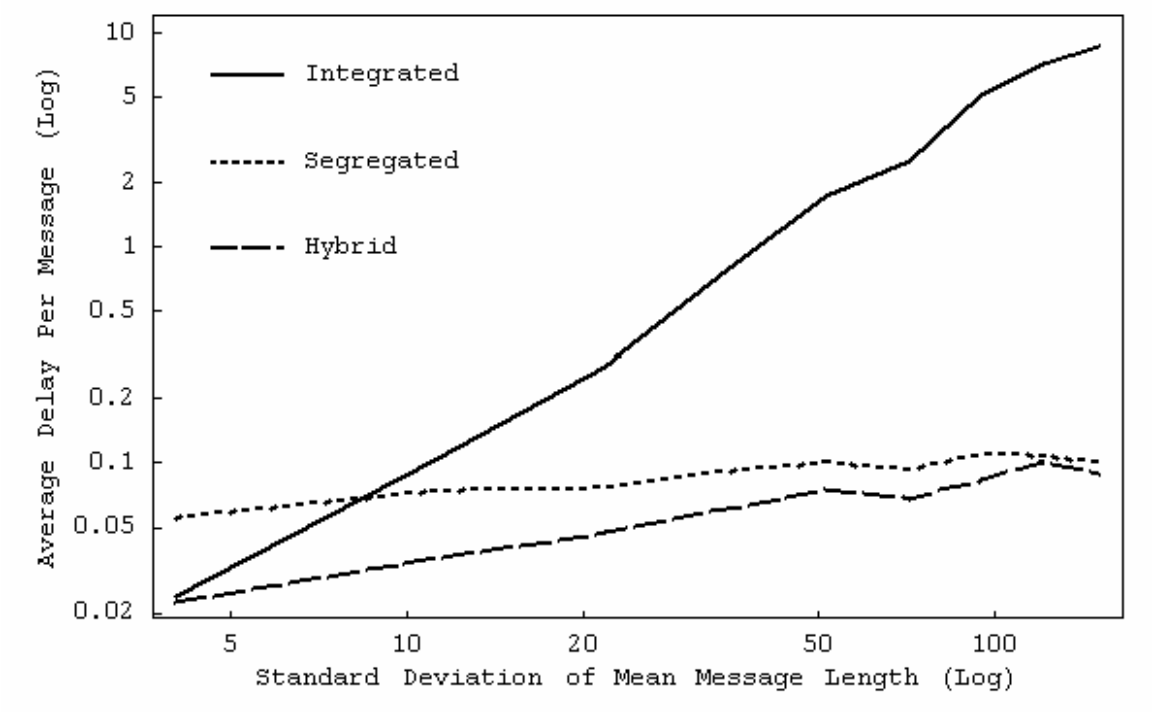

Figure 5.7: Plot of the simulation results for the M/LN/1 queue

5.5.4 Results for LN/LN/1, P/LN/1 and P/P/1 queues

	1	2	3	4	5	6	7	8	9
λ_1	328.5	328.5	328.5	328.5	328.5	328.5	328.5	328.5	328.5
$1/\mu_1$	1	1	1	1	1	1	1	1	1
λ_2	203.4	135.6	101.7	81.36	67.8	58.11	50.85	45.2	40.68
$1/\mu_2$	2	3	4	5	6	7	8	9	10
λ_3	49.73	28.41	19.89	14.21	10.47	7.956	6.416	5.376	4.52
$1/\mu_3$	4	7	10	14	19	25	31	37	44
λ_4	11.52	5.236	2.743	1.694	1.152	0.8229	0.6128	0.476	0.3789
$1/\mu_4$	5	11	21	34	50	70	94	121	152
λ_5	75.09	27.66	12.82	7.008	4.344	2.872	2.014	1.468	1.107
$1/\mu_5$	7	19	41	75	121	183	261	358	475
λ_6	50.4	14.18	5.815	2.908	1.643	1.015	0.668	0.4619	0.3321
$1/\mu_6$	9	32	78	156	276	447	679	982	1366
σ_L	2.525	6.22	10.69	15.77	21.36	27.47	34.02	41.01	48.43

Table 5.6: Input data for the G/G/1 queue analysis

	σ_L	T_{int}	T_{seg}	T_{hyb}	Optimal Hybrid Grouping
1	2.5249	0.096378	0.30638	0.096378	{{1, 2, 3, 4, 5, 6}}
2	6.2201	0.2745	0.40804	0.18166	{{1, 2}, {3, 4, 5, 6}}
3	10.687	0.61042	0.46699	0.25193	{{1, 2}, {3, 4, 5, 6}}
4	15.771	1.1561	0.50536	0.29381	{{1}, {2, 3}, {4, 5, 6}}
5	21.359	1.9564	0.53194	0.32166	{{1}, {2, 3}, {4, 5, 6}}
6	27.474	3.073	0.55142	0.34652	{{1}, {2, 3}, {4, 5, 6}}
7	34.023	4.5488	0.56591	0.36705	{{1}, {2, 3}, {4, 5, 6}}
8	41.015	6.4437	0.57715	0.38457	{{1}, {2, 3}, {4, 5, 6}}
9	48.432	8.8101	0.58632	0.40204	{{1}, {2, 3}, {4, 5, 6}}

Table 5.7: Analytical results for the LN/LN/1 queue

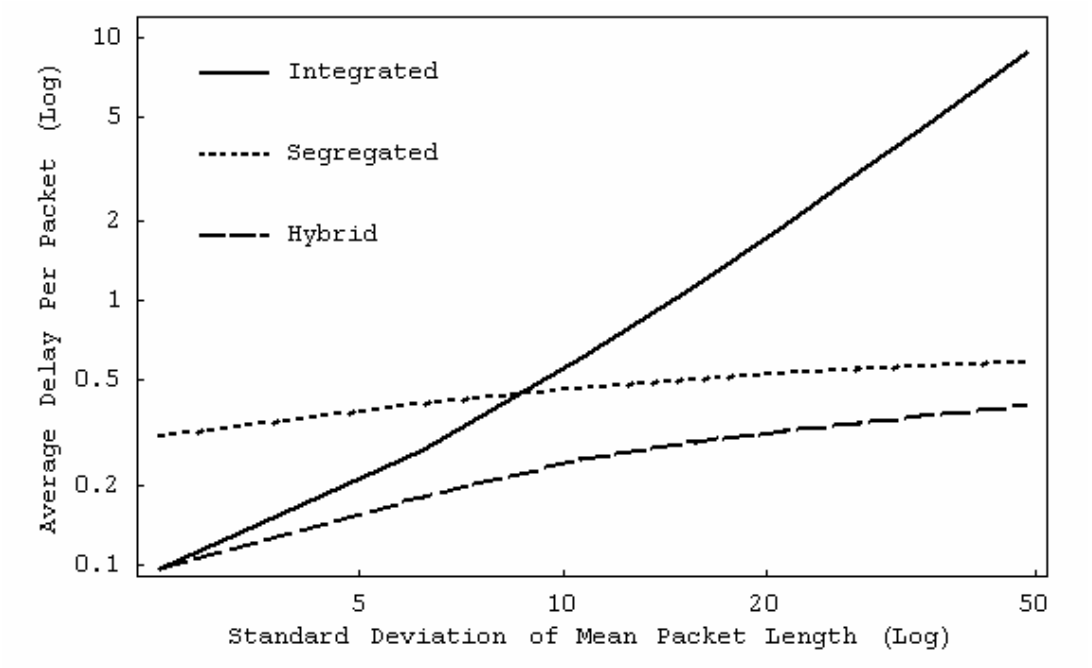

Figure 5.8: Plot of the analytical results of the LN/LN/1 queue

	σ_L	T_{int}	T_{seg}	T_{hyb}
1	2.5249	0.092437	0.29004	0.093437
2	6.2201	0.22862	0.40707	0.1607
3	10.687	0.63181	0.41873	0.24849
4	15.771	1.0707	0.48362	0.28155
5	21.359	1.3689	0.52289	0.25244
6	27.474	2.2097	0.43986	0.28806
7	34.023	2.2531	0.41339	0.25026
8	41.015	12.605	0.77406	0.62016
9	48.432	4.2216	0.44115	0.29038

Table 5.8: Simulation results for the LN/LN/1 queue

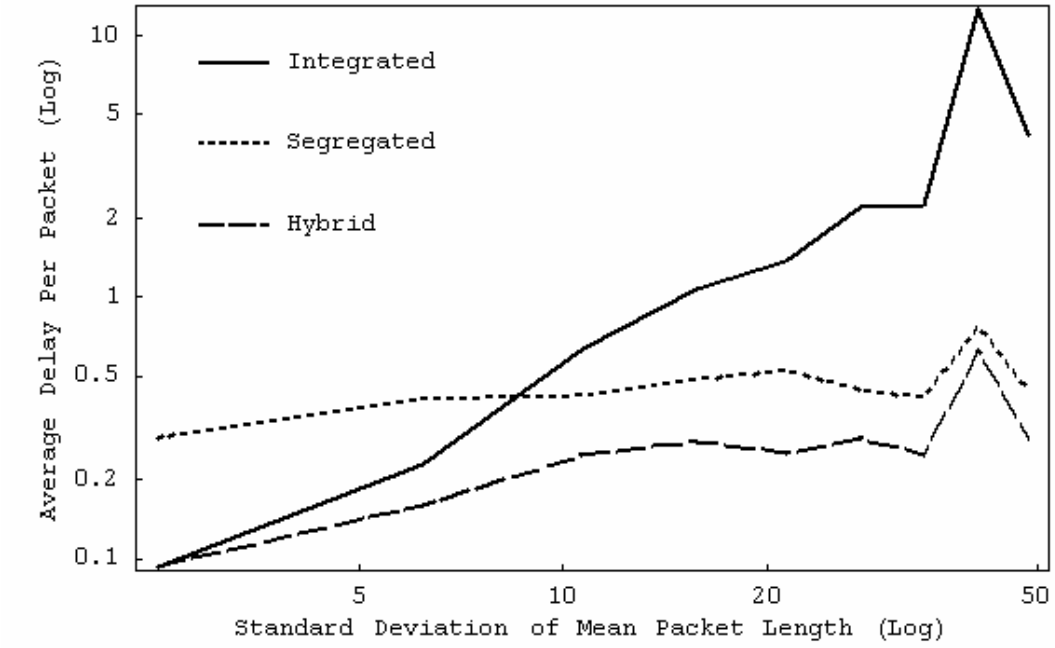

Figure 5.9: Plot of the simulation results of the LN/LN/1 queue

	σ_L	T_{int}	T_{seg}	T_{hyb}
1	2.5249	0.16229	0.67568	0.16329
2	6.2201	0.43107	0.81858	0.35264
3	10.687	0.9627	0.88311	0.4961
4	15.771	2.1535	1.3582	0.76023
5	21.359	2.3372	0.95753	0.63516
6	27.474	3.673	1.0831	0.67398
7	34.023	6.8425	1.1113	0.81054
8	41.015	8.6295	1.0547	0.70725
9	48.432	8.3994	1.0551	0.73008

Table 5.9: Simulation results for the P/LN/1 queue

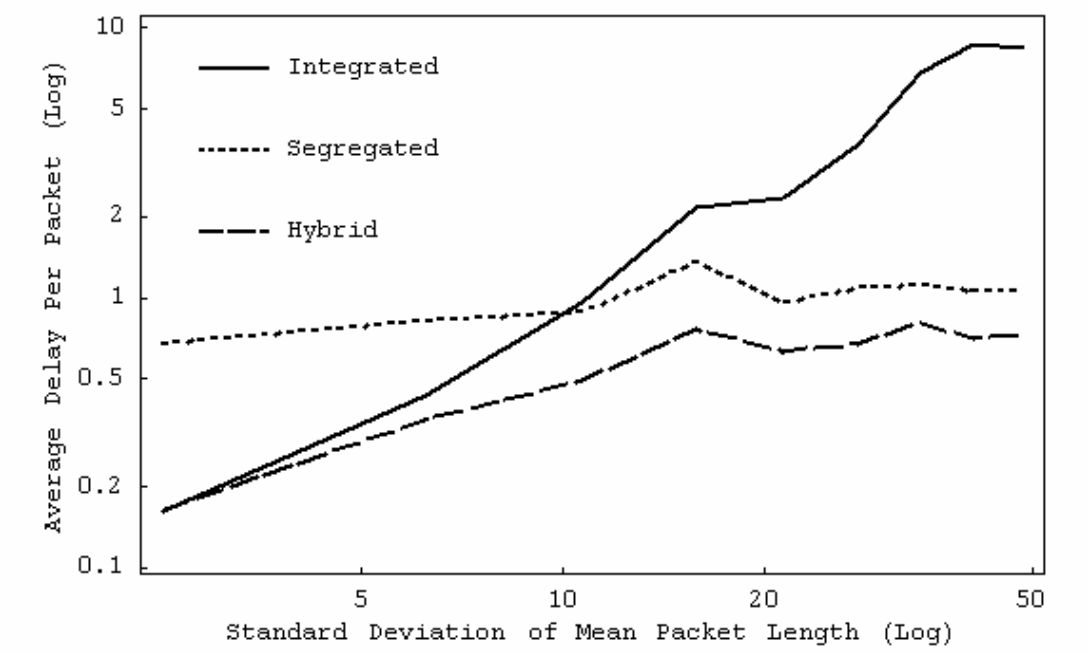

Figure 5.10: Plot of the simulation results of the P/LN/1 queue

	σ_L	T_{int}	T_{seg}	T_{hyb}
1	2.5249	0.12931	0.59253	0.13031
2	6.2201	0.28781	0.63996	0.25893
3	10.687	0.73994	0.84915	0.39208
4	15.771	0.79832	0.6426	0.39201
5	21.359	2.0319	0.72506	0.47064
6	27.474	3.6686	0.89774	0.53772
7	34.023	2.8193	0.65018	0.50157
8	41.015	4.0233	0.59034	0.46674
9	48.432	2.8425	0.62505	0.44928

Table 5.10: Simulation results for the P/P/1 queue

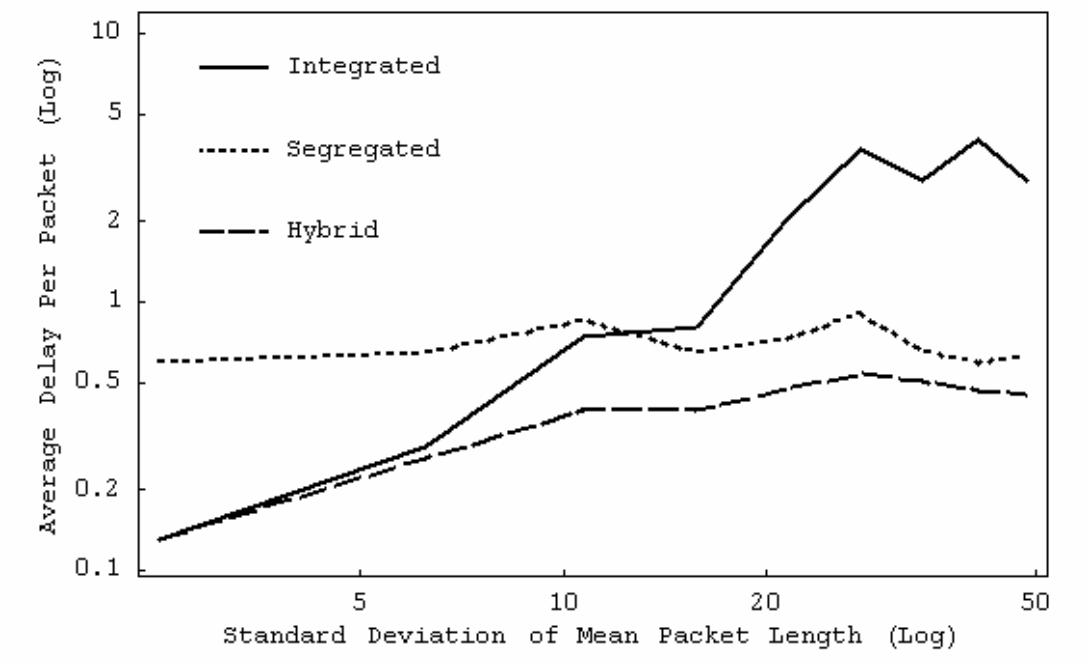

Figure 5.11: Plot of the simulation results of the P/P/1 queue

5.5.5 Discussion of the Results

By examining the analytical and simulation results, it can be seen that at low values of standard deviation, the integrated approach results in a lower delay. On the other hand, the segregation approach results in a lower delay at higher values of standard deviation. Since hybrid integration can either be fully segregated or fully

integrated, it always matches the lowest delay of either one. For example, Table 5.7 and Figure 5.8 show that hybrid integration is equivalent to full integration in dataset 1. In other datasets, it can be seen that there are hybrid grouping arrangements that yield lower delay than either segregation or integration. The results show significant improvement in the delay. Table 5.5 shows a close match between the analytically calculated delay values and actual delay values obtained from simulations. This demonstrates the accuracy of the developed analytical models.

5.6 Conclusions

This chapter has introduced the new hybrid integration approach for queue management in packet switched networks. In this approach, flows are grouped in a certain arrangement that yields the lowest possible queueing delay. As special cases, the optimum grouping arrangement can be complete segregation or complete integration of the flows, which are the cases studied in the literature. In order to determine the optimum grouping, a mathematical model is required for calculating the delay for each arrangement. Generally, it is required to calculate the delay for all possible arrangements to decide the optimum one. However, several methods have been proposed in this chapter that can significantly reduce the computation requirements. In addition, some heuristic methods have been proposed for determining approximate or near optimum arrangements when analytical models are not available. In this chapter, analytical models have been developed for M/G/1 queues and G/G/1 queues to calculate the delay for the hybrid integrated system. These models have used heavy tailed distributions including the lognormal and Pareto distributions. The hybrid integration approach has been evaluated using the developed

analytical models and using simulation experiments. The results of both have shown great improvement in the average delay. In addition, the results have shown that hybrid integration either outperforms or at least matches the lowest delay achievable by either of the segregation or integration approaches.

Chapter 6. Quality of Service Techniques

Abstract: This chapter reviews the three main Quality of Service techniques: Integrated Services (IntServ), Differentiated Services (DiffServ) and the Multi Protocol Label Switching (MPLS). The main features of each technique are discussed. A brief comparison between IntServ and DiffServ is presented. The purpose of this chapter is to provide the background necessary to understand the new results contributed by this dissertation which are presented in the following two chapters.

Previous chapters have studied performance enhancement of networks without priorities. The focus has been on the improvement of the overall network performance while maximizing the utilization of resources. As previously mentioned in Chapter 1, it is essential for data networks to be able to explicitly specify priorities based on the application or user requirements. Different Quality of Service techniques have already been developed for this purpose. The findings presented in previous chapters are used to characterize and improve these techniques in the next part of this dissertation.

This chapter provides a brief overview of the three main edge-to-edge Quality of Service techniques currently available for the Internet: the Integrated Services architecture (IntServ), the Differentiated Services architecture (DiffServ) and the Multi Protocol Label Switching architecture (MPLS). These techniques share some common concepts. The network is divided into edge routers and core routers. Edge routers are considered the gateway to the network. They are connected directly to the

end user local area networks. Core routers form the backbone of the network. They provide the connectivity between multiple large scale networks [5]. A network model for IntServ, DiffServ and MPLS is shown in Figure 6.1.

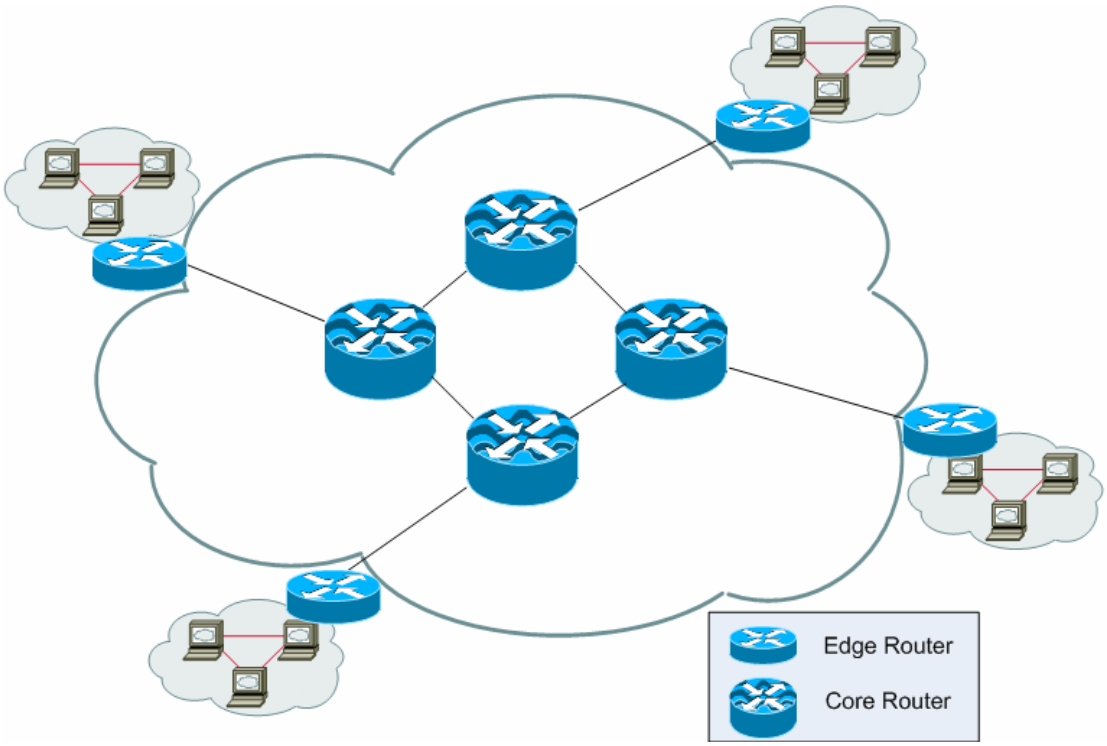

Figure 6.1: Network model for IntServ, DiffServ and MPLS

6.1 Integrated Services

The Integrated Service architecture [40] was developed by the IETF in the early 1990s to support the requirement of real-time applications. It is one of the first attempts to support QoS over the Internet [7]. IntServ is considered as a *micro-level QoS* model because it operates on a per-flow basis. It is also considered as a *hard QoS* model because it requires end-to-end reservation of resources on each router along the path prior to data transmission [41]. Additionally, each router has to keep track of active flows during the lifetime of the connections.

6.1.1 Application Classes

IntServ classifies applications into three main categories [6]:

- Elastic applications which can tolerate a wide range of data rates, delays and packet losses. Examples include email, File Transfer Protocol (FTP), Domain Name Service (DNS) [5].
- Tolerant real-time applications which can tolerate some packet loss and limited delay. Multimedia streaming applications and internet gaming applications fit into this class.
- Intolerant real-time applications such as voice and video conferencing. They have stringent bandwidth, delay and jitter requirements.

6.1.2 Service Classes

The IntServ architecture defines two service classes to accommodate the different application classes. Besides the default best effort behavior, which is suitable for elastic applications, the following service classes are defined [6]:

- Controlled Load service (CL) class. This class emulates the behavior of a network with no or light load. It is intended to be better than best effort but can still have some occasional delay or packet loss. This service class is suitable for tolerant real-time applications.
- Guaranteed Service (GS) class. This class provides strict bounds on bandwidth and delay. It is intended for intolerant real-time applications.

6.1.3 Basic Operation

The operation of data transmission over an IntServ network can be summarized as follows [5, 7]:

- Resource requirements are determined by the application and described as *flow specification*.
- A Routing protocol is used by the network to find the best available path to destination. The IntServ architecture does not mandate a specific routing protocol. It relies on the routing protocol to select the next hop.
- The Resource Reservation Protocol (RSVP) is used to perform the signaling and admission control functions. The source sends control messages to identify flow specifications and to collect information about the routers along the path. Multiple receivers can be specified. Each receiver responds with control messages identifying its QoS requirements. Depending on the flow requirements and resource availability, the connection request is either admitted or denied.
- Routers along the path keep track of all active flows. Each receiver sends a renew message periodically in order to maintain the reservation.

6.1.4 Outstanding Issues

IntServ is a significant breakthrough in providing QoS over IP networks. It is capable of providing strict bounds on requested resources for critical applications. In IntServ, resource assurance can be provided with a high degree of precision. Individual flow control obviates the need for complex queue management mechanisms [5]. In addition, some of the techniques developed for IntServ have inspired the development of other QoS mechanisms, such as DiffServ and MPLS [7].

Despite these features, IntServ was soon realized to be unsuitable for large scale networks, such as the Internet. In such networks, scalability is an important objective. Maintaining a per-flow status at each router requires a substantial amount of memory and processing power, especially at core routers where the number of concurrent connections is in the range of hundreds of thousands over very high speed links such as OC-48. Studies have shown that even for OC-3 links, the number of concurrent connections can reach 250,000 in the core routers [6]. Reservation setup is more suitable for long lasting connections, such as multimedia conferencing, for which IntServ was initially designed. However, the short-lived nature of web applications, which dominate the current Internet, made resource reservation a prohibitive overhead [7]. Instead of the significant breakthrough brought by IntServ, the IETF community decided that it is more effective to design a QoS model that offers gradual improvement of the best effort service. This decision resulted in the development of the Differentiated Services architecture.

6.2 Differentiated Services

The Differentiated Services architecture (DiffServ) [42] was proposed by the IETF in the late 1990s as a viable alternative to IntServ. DiffServ is a *macro-level* or *coarse-grained* approach which provides per-aggregate instead of per-flow service differentiation. The idea behind DiffServ development was to provide an incremental improvement over the best effort service, rather than a radical advancement as IntServ aimed to achieve.

In DiffServ, traffic is classified into a small number of forwarding classes. Different treatment can be defined for each class, which is also known as a Per-Hop

Behavior (PHB). The PHB is applied to all flows within the class without a need to identify individual flows. Therefore, DiffServ is not limited by the number of flows. The signaling phase is not required in DiffServ. No resource reservation is performed [5, 7].

DiffServ pushes the complexity to the network edge, where the number of flows is manageable. Edge routers perform two main functions, traffic classification and traffic conditioning, also known as admission control. Both functions are governed by the Service Level Agreement (SLA). Generally, packets conforming to the SLA are considered in-profile and packets exceeding SLA are considered out-profile. The classification process examines incoming packets at the ingress router against the rules defined by the SLA. Packets are assigned the appropriate class (EF—Expedited Forwarding—or one of the AF—Assured Forwarding—classes) by marking them with the corresponding Differentiated Services Code Point (DSCP or DS field) field value. The conditioning process ensures that user flows stay within the SLA. Depending on flow characteristics and network conditions, out-profile packets may be marked with higher drop precedence, delayed in the queue or dropped. A commonly used admission control technique is the Token Bucket algorithm [5, 7].

With complexity pushed to edge routers, core routers have simpler functions in DiffServ. PHB actions are defined for each class. The router merely checks the DSCP field and performs the appropriate action. Queue management and scheduling techniques are used at both edge and core routers.

6.2.1 Forwarding Classes

DiffServ defines the following main forwarding classes or Per-Hop Behaviors [7, 41, 43]:

- Expedited Forwarding (EF): This is the top priority class. Packets marked with expedited forwarding should be forwarded as soon as they arrive regardless of the current state of the router. They are expected to encounter the lowest possible delay and loss.
- Assured Forwarding (AF): AF defines a group of four classes with three drop precedence levels. At least two drop priorities must be implemented at each DiffServ router. To avoid packet reordering, edge routers should not assign different AF classes to packets from the same flow.
- Best Effort (BE): This is the default lowest priority class.

DiffServ specifications do not mandate specific mechanisms for implementing PHBs. Instead, they define a set of requirements to be accomplished for each PHB class.

To distinguish different classes, DiffServ utilizes 6 bits of the 8-bit Type Of Service (TOS) field of the IP header [44]. This allows up to 64 possible classes. In DiffServ, this field is referred to as Differentiated Service Code Point (DSCP) or forwarding class. Some DSCP values are reserved for different purposes.

6.2.2 Basic Operation

DiffServ operation can be summarized as follows [41]:

- The source node initiates data transmission and sends a request to the edge or first-hop router.

- The edge router contacts the Bandwidth Broker (BB) router. The bandwidth broker is responsible of managing network QoS resources within the DiffServ domain [6]. The bandwidth broker determines if there are sufficient resources and accepts or rejects the connection accordingly. This process is called “admission control”.
- Provided that the connection is accepted, the DSCP field is marked with the appropriate PHB class number.
- The edge router performs the necessary admission control and traffic conditioning, shaping and policing (e.g. marking incompliant packets with higher drop precedence) before forwarding packets to the network core.
- Packets are forwarded through the core routers. Core routers forward packets based on the value of the DSCP field.

6.3 Multi-Protocol Label Switching

Multi-Protocol Label Switching (MPLS) [45] is a label switching protocol that provides a robust switching mechanism over the network layer. Although MPLS is not bound to a specific switching protocol, it can utilize link layer switching protocols such as ATM and Frame Relay. Similar to DiffServ, MPLS aims to push the complexity to the edge of the network to enable faster performance at the core [7]. In the MPLS context, the burden of looking up long packet headers to determine the next hop in the router is eliminated for core routers. The route is constructed upon connection setup. This route can then be encoded in short labels (tags) that are attached to each IP packet. MPLS Label Switched Routers (LSRs) recognize MPLS

labels and use them to select the outbound port based on their switching tables. Thus, MPLS can be thought of as a virtual circuit mechanism for IP.

MPLS can also perform traffic engineering by explicitly specifying non shortest-path routes to balance load on network links and reduce congestion. In addition, MPLS can provide QoS by providing bandwidth reservation for specific Label Switched Paths (LSPs). DiffServ can be combined with MPLS to provide explicit paths for different service classes [43]. IntServ can use MPLS to explicitly specify the reserved path. Both IntServ and DiffServ can be combined with MPLS to provide more intelligent QoS techniques [46].

6.3.1 Forward Equivalent Classes

In MPLS, each Forward Equivalent Class (FEC) defines packets that are forwarded on the same path. FEC is assigned to the packet once by the edge router (LER). FEC assignment can be done based on different criteria. It can be assigned to all packets having the same destination, the same source and destination pair or with the same TCP/UDP port number. FECs are only recognized within adjacent routers such as LER and the immediate LSR or two adjacent LSRs [43].

6.3.2 Label Distribution

Label distribution is a set of procedures defined to exchange FEC/label bindings between LSRs. Labels must be exchanged so that adjacent routers can understand the meaning of the labels. Label distribution is done in one of the following methods [6]:

- Upstream allocation: label distribution is performed by the source node (with respect to the dataflow)
- Downstream allocation: label distribution is performed by the receiving or destination node.
- Local allocation: the label/FEC binding is performed by the local LSR. This is done either in advance (i.e., prior to connection) or when the control information is received from the source or destination node (i.e., on-demand).

6.3.3 Basic Operation

Routing in MPLS is performed as follows [43]:

- A Packet enters the LER and is assigned an FEC.
- The LER adds a label to the packet and forward it the next LSR.
- Each LSR reads the label, performs a table lookup for a label/FEC binding, swaps the label and forwards the packet to the next LSR through the appropriate port.

6.4 Comparison

Table 6.1 summarizes the main differences between IntServ and DiffServ architectures.

	IntServ	**DiffServ**
Coordination for Service Differentiation	End-to-End	Local (per-hop)
Scope of Service	A Unicast or Multicast path	Anywhere in a network or in specific paths
Scalability	Limited by the number of flows	Limited by the number of classes of service
Network Accounting	Based on flow characteristics and QoS requirement	Based on class usage
Network Management	Similar to circuit switching networks	Similar to existing IP networks
Inter-domain deployment	Multilateral agreements	Bilateral agreements
Granularity of Service differentiation	Individual flow	Aggregate of flows
State in routers (e.g. scheduling, buffer management)	Per flow	Per aggregate
Traffic classification basis	Several header fields	DS field
Type of service differentiation	Deterministic or statistical guarantees	Absolute or relative assurance
Admission Control	Required	Required for absolute differentiation
Signaling protocol	Required (RSVP)	Not required for relative schemes

Table 6.1: Comparison between DiffServ and IntServ [47]

6.5 Summary

This chapter has provided a brief overview of the three main QoS techniques: IntServ, DiffServ and MPLS. While IntServ provides strict QoS assurance on a per-flow level, it is unsuitable for deployment on large scale networks. DiffServ is a more scalable solution that provides differentiation on the aggregate level. MPLS is a label switching technique that is used for speeding up the routing operation and can also be used for traffic engineering and providing QoS when combined with IntServ or DiffServ.

The queueing analyses and optimization methods developed in this dissertation can be used to improve both IntServ and DiffServ. For IntServ, the segregation / integration analyses are used to derive a fair and more accurate pricing model for flow reservation in Chapter 7. In Chapter 8, the concepts of segregation and hybrid integration are used to develop a new mechanism for solving fairness problems and improving the performance in terms of the delay, throughput and packet loss in DiffServ.

Chapter 7. Resource Based Pricing Model for Integrated Services Architecture

Abstract: This chapter addresses the impact of resource assurance provided by IntServ resource reservation on the overall bandwidth requirements and proposes a scheme whereby a service provider can develop compensatory and fair prices for customers with varying QoS under a wide variety of ambient traffic scenarios. The material of this chapter has been published in [48].

As mentioned in the previous chapters, different techniques have been designed for providing quality of service in the Internet. However, scant attention has been paid toward understanding the resource cost associated with the implementation of QoS techniques.

Different pricing approaches have been studied in the literature. Among these approaches are flat-rate pricing [49, 50], usage-based pricing [51], priority based pricing [52, 53] and congestion-based pricing [54, 55]. The approach proposed in this chapter aims to make pricing more relevant to the actual resource usage. Further, it addresses how pricing should vary as the QoS is increased on demand.

To achieve a closed form solution, the analysis performed in this chapter assumes an M/M/1 queue. Partial analysis is provided for an M/G/1 queue in section 7.1. This chapter addresses the Integrated Services architecture. In particular, it focuses on the effect of exclusive bandwidth reservation for flows within the same service class. IntServ uses the Resource Reservation (RSVP) protocol for allocating

bandwidth for each flow upon connection establishment [5]. Whether the remaining bandwidth out of the common pool is shared or divided, the other flows will be affected by this reservation. As a result, in order to maintain the same level of performance and pricing for the other flows, the service provider should either reduce the number of flows, thus reducing the revenue, or increase the total bandwidth increasing the cost. This dissertation proposes a pricing scheme that fairly reflects the cost associated with the bandwidth reservation.

7.1 Mathematical Background

This section provides proofs and derivations required in this chapter.

7.1.1 Service Time Distribution of a Queue Composed of Individual Queues

Suppose X is a random variable representing the packet size of the shared combined queue. X_i (i=1,…,n) represents the packet size of the individual queue i. The following relation applies:

$$\begin{aligned}\Pr[X = x] &= \frac{\lambda_1}{\lambda}\Pr[X_1 = x] \\ &+ \frac{\lambda_2}{\lambda}\Pr[X_2 = x] + \cdots + \frac{\lambda_n}{\lambda}\Pr[X_n = x]\end{aligned} \tag{7.1}$$

Since $X_1, X_2, \ldots, X_n$ have the same distribution and have identical mean and statistical properties. Then:

$$\Pr[X_1 = x] = \Pr[X_2 = x] = \cdots = \Pr[X_n = x] \tag{7.2}$$

Substituting in (7.1):

$$\Pr[X = x] = \left(\sum_{j=1}^{n} \frac{\lambda_j}{\lambda} \right) \Pr[X_1 = x] \tag{7.3}$$

From (7.3), it can be stated that:

$$\Pr[X = x] = \Pr[X_i = x], \qquad i = 1, \cdots, n \tag{7.4}$$

i.e., combining multiple flows of traffic with identical packet size distributions result in a system with the same packet size distribution. Since the service time is simply the packet size divided by the channel capacity which is constant. The same can be said about the distribution of the service time.

7.1.2 Delay Analysis for M/G/1 Queueing Systems

The service time is determined from the packet size divided by the channel capacity. Thus, if X represents the packet size, then $S = X/C$ is the service time for the combined queue and $S_i = X_i/(C/n)$ is the service time for the individual queue formed by equally dividing the bandwidth among flows. Using the proof in 7.1.1, the distributions X and X_i are identical.

From basic statistics:

$$E[aX] = a\,E[X] \tag{7.5}$$

$$\text{var}[aX] = a^2\,\text{var}[X] \tag{7.6}$$

Applying to the service time distributions:

$$E\left[n\frac{X}{C} \right] = n\,E\left[\frac{X}{C} \right] \tag{7.7}$$

$$\text{var}\left[n\frac{X}{C} \right] = n^2\,\text{var}\left[\frac{X}{C} \right] \tag{7.8}$$

Therefore:

$$\mathrm{var}\left[S_i\right] = n^2\,\mathrm{var}\left[S\right], \quad i = 1, \cdots, n \tag{7.9}$$

The variance of the service time for individual queues is n^2 times greater than the combined queue when the bandwidth is equally split between them. This result is independent of the distribution of the packet size (or the service time).

Let σ_S^2 denote the variance of the service time of the combined queue and σ_{Si}^2 denote the variance of the service time of any of the individual queues *i*. Using the P–K formula [16], the delay *T* for the combined M/G/1 queue can be calculated as:

$$T = \frac{1}{\mu C} + \frac{\rho^2 + \lambda^2 \sigma_S^2}{2\lambda\left(1-\rho\right)} \tag{7.10}$$

For all individual queues *i*, the delay T_i is calculated as:

$$T_i = \frac{1}{\mu C/n} + \frac{\rho^2 + \dfrac{\lambda^2}{n^2}\sigma_{Si}^2}{2\dfrac{\lambda}{n}\left(1-\rho\right)} \tag{7.11}$$

From (7.9):

$$\sigma_{Si}^2 = n^2 \sigma_S^2 \tag{7.12}$$

Substituting in (7.11):

$$\begin{aligned} T_i &= n\left(\frac{1}{\mu C} + \frac{\rho^2 + \lambda^2 \sigma_{Si}^2}{2\lambda\left(1-\rho\right)}\right) \\ &= n\,T \end{aligned} \tag{7.13}$$

This shows that the mean delay for the individual queues is *n* times greater than the mean delay for the combined queue if the total bandwidth is equally shared and all packet sizes have the same mean. This result is valid for M/G/1 queues

regardless of the distribution of the packet sizes (or service times) provided that the service time variance has a finite value.

7.1.3 Delay and Variance Calculation for M/M/1 Queues

Kleinrock [16] has shown that the k-th moment of the system delay $D^{(k)}$ for the M/G/1 queueing system is calculated as:

$$D^{(k)} = \sum_{i=0}^{k} w^{(k-i)} b^{(i)} \tag{7.14}$$

where:

$$w^{(k)} = \frac{\lambda}{1-\rho} \sum_{i=1}^{k} \binom{k}{i} \frac{b^{(i+1)}}{i+1} w^{(k-i)} \tag{7.15}$$

The notation $b^{(i)}$ denotes the i-th moment of b, w denotes the waiting time and D denotes the system delay which equals the waiting time plus the service time.

For the M/M/1 system, the service time is exponentially distributed. Thus, the first three moments are given as

$$b^{(0)} = 1 \tag{7.16}$$

$$b^{(1)} = \frac{1}{\mu C} \tag{7.17}$$

$$b^{(2)} = \frac{2}{\mu^2 C^2} \tag{7.18}$$

$$b^{(3)} = \frac{6}{\mu^3 C^3} \tag{7.19}$$

Using these equations, the variance of the system delay can be calculated as follows:

$$w^{(1)} = \overline{w} = \frac{\lambda}{\mu^2 C^2 (1-\rho)} \tag{7.20}$$

$$w^{(2)} = \frac{2\lambda^2}{\mu^4 C^4 (1-\rho)^2} + \frac{2\lambda}{\mu^2 (1-\rho)} \tag{7.21}$$

$$D^{(1)} = \frac{1}{\mu C} + \frac{\lambda}{\mu^2 (1-\rho)} \tag{7.22}$$

$$D^{(2)} = \frac{2}{\mu^2 C^2} + \frac{2\lambda^2}{\mu^4 C^4 (1-\rho)^2} + \frac{4\lambda}{\mu^2 C^2 (1-\rho)} \tag{7.23}$$

$$\sigma_D^2 = D^{(2)} - \left(D^{(1)}\right)^2 = \frac{1}{\mu^2 C^2 (1-\rho)^2} \tag{7.24}$$

7.2 Delay Analysis

Suppose homogeneous traffic served by an M/M/1 queueing system is split into *n* flows on a random basis. Each flow will have the same mean service time and will be identically distributed with Poisson arrivals [56] and exponential service times (subsection 7.1.1). In particular, if the distribution of the overall traffic is such that each arrival has an identical probability of falling in any one of the *n* flows, the utilization of all sub-channels will be the same. As shown in Chapters 4 and 5, the segregation of traffic into separate channels improves the weighted mean delay when the different flows have disparate mean packet sizes [30, 31]. This is not the case, however, when the traffic is homogenous [18].

The split of traffic into *n* sub-channels can be formalized as follows. Note that the suffixes 1,2,…,*n* represent each of the *n* channels into which traffic is distributed with identical probability. Thus,

$$\rho_1 = \rho_2 = \cdots = \rho_n \tag{7.25}$$

$$1/\mu_1 = 1/\mu_2 = \cdots = 1/\mu_n \tag{7.26}$$

$$\lambda_1 = \lambda_2 = \cdots = \lambda_n = \frac{\lambda}{n} \tag{7.27}$$

$$C_1 = C_2 = \cdots = C_n = \frac{C}{n} \tag{7.28}$$

The mean delay for the combined (integrated) system can be calculated using (2.8), i.e.:

$$T = \frac{\rho/\lambda}{1-\rho} \tag{7.29}$$

For each sub-channel *i*, the mean delay is given by:

$$T_i = \frac{\rho/\lambda_i}{1-\rho} = n\frac{\rho/\lambda}{1-\rho}, \quad i = 1, \cdots, n \tag{7.30}$$

Comparing (7.29) and (7.30), it can be seen that:

$$T_i = nT, \qquad i = 1, \cdots, n \tag{7.31}$$

In other words, the mean delay of *n* identically distributed M/M/1 systems is *n* times that of a single M/M/1 system serving the summation of the traffic using a bandwidth that is the sum of all the individual channels. This result is also valid for M/G/1 queues as has been shown in subsection 7.1.2

7.3 Jitter Analysis

Jitter plays an important role in the quality of real-time traffic, such as voice or video communications. The mean opinion score (MOS) of telephony traffic, for example, is impacted by not only the fixed delay that the transmission system would impose, but also by the variability of delay, namely, jitter [57]. The jitter can be

estimated by the variance of the delay [58, 59]. In this part, the variances of the mean delays for the single channel and the *n* sub-channels are compared.

The variance of the delay σ_D^2 for the M/M/1 queue is calculated as:

$$\sigma_D^2 = \frac{1}{\mu^2 C^2 (1-\rho)^2} \tag{7.32}$$

The derivation of (7.32) is provided in subsection 7.1.3 .

Using (7.32), the variance of the delay for each sub-channel *i* is calculated as:

$$\begin{aligned} \sigma_{Di}^2 &= \frac{1}{\mu^2 (C/n)^2 (1-\rho)^2} \\ &= n^2 \frac{1}{\mu^2 C^2 (1-\rho)^2} \qquad i = 1, \cdots, n \end{aligned} \tag{7.33}$$

The variance of the combined system is calculated from (7.32).

Comparing (7.32) and (7.33) :

$$\sigma_{Di}^2 = n^2 \sigma_D^2, \quad i = 1, \cdots, n \tag{7.34}$$

This shows that the variance, and hence the jitter, is lower by a factor of n^2 in the fully shared system compared to that in the system with separate bandwidth reservation for each of the *n* flows.

7.4 Compensatory Pricing for Guaranteed Quality of Service

The discussion in the previous sections shows that there is a performance penalty imposed by reserving exclusive bandwidth for each flow of traffic. Without such exclusive allocation of bandwidths for specific flows, the service provider is able to provide an average delay that is equal to $1/n$ times and a jitter that is $1/n^2$ times that of the system with equal exclusive bandwidth allocation to each flow.

7.4.1 Individual Resource Requirements

This part provides the derivation required in order to estimate the resources consumed in individually reserved channels against that in a shared bandwidth system.

For the combined system, (7.29) can be rewritten as:

$$T = \frac{1}{\mu C - \lambda} \tag{7.35}$$

where $\lambda=\lambda_1+\lambda_2+\ldots+\lambda_n$ and T is the mean delay per packet. Suppose it is required to achieve the same value of T for an individual flow in an exclusive bandwidth-allocated system. It is required to determine the bandwidth $C_{\{1\}}$ that satisfies:

$$\frac{1}{\mu C_{\{1\}} - \lambda/n} = \frac{1}{\mu C - \lambda} \tag{7.36}$$

where λ/n is the arrival rate of any individual flow.

Solving (7.36) yields:

$$C_{\{1\}} = C - \frac{\lambda}{\mu}\left(\frac{n-1}{n}\right) = C\left(1 - \rho\left(\frac{n-1}{n}\right)\right) \tag{7.37}$$

Dividing both sides of (7.37) by C yields:

$$\frac{C_{\{1\}}}{C} = 1 - \rho\left(\frac{n-1}{n}\right) \tag{7.38}$$

The stability of any queueing system implies that ρ is always less than 1, yielding:

$$\frac{C_{\{1\}}}{C} > 1 - \left(\frac{n-1}{n}\right) \tag{7.39}$$

which is equivalent to:

$$C_{\{1\}} > \frac{C}{n} \tag{7.40}$$

The inequality in (7.40) is a very important result. It shows that the capacity required for each individual flow in a bandwidth-reserved system serving *n* flows is always more than 1/*n* times the total available capacity if it is required to achieve the same delay as for a combined, shared-bandwidth system serving the same number of flows with the same total capacity. The exact amount of capacity required can be calculated from (7.37).

Applying the same analysis to equate the variance (jitter) of the reserved bandwidth system with that of the bandwidth-shared system, (7.32) can be rewritten as:

$$\sigma_D^2 = \frac{1}{(\mu C - \lambda)^2} \tag{7.41}$$

The value of $C_{\{1\}}$ required to equate the variance should satisfy:

$$\frac{1}{\left(\mu C_{\{1\}} - \lambda/n\right)^2} = \frac{1}{(\mu C - \lambda)^2} \tag{7.42}$$

Solving for $C_{\{1\}}$ yields:

$$C_{\{1\}} = C\left(1 - \rho\left(\frac{n-1}{n}\right)\right) \tag{7.43}$$

which is the same value obtained in (7.37). Equations (7.37) and (7.43) together point out the fact that the additional capacity needed to equalize the mean delays is both necessary and sufficient to equalize the variances of the delays of the two systems as well. Note that this is generally not true for systems other than M/M/n.

Figure 7.1 shows channel capacity required by individual flows in order to maintain the same delay in the shared-bandwidth system divided by C/n, where C is the total channel capacity for the system and n is the number of flows. The plot is done for different system loads or utilization factors. The utilization factor is for the system before allocation. It can be observed from Figure 7.1 that the capacity required for each flow to keep the same delay before allocation is higher if the utilization of the system before allocation is lower. This is because the delay is inversely related to the utilization. Therefore, more bandwidth is needed for a flow to match a delay of a lightly loaded shared-bandwidth system. This is the case for the Controlled Load (CL) service class of IntServ, which emulates the behavior of a lightly loaded best effort system [5].

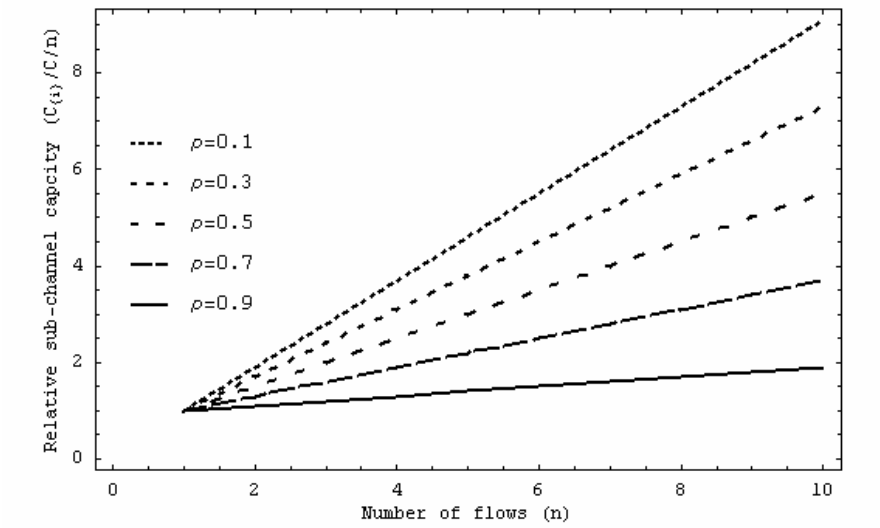

Figure 7.1: Relative capacity requirement for n sub-channels with different utilizations

7.4.2 Additional Cost Requirements

Whenever a customer or a flow of traffic in a shared system requests an exclusive bandwidth, it is expected that the other customers or flows will be affected.

It can be seen from (7.40) that the separated flow takes more than simply its equal share of the total bandwidth. Thus, the other flows are likely to encounter more delay. In order to maintain the delay for the other flows without excluding some of them, the provider must increase the capacity. The flow requesting exclusive bandwidth should be charged for that increment.

In the M/M/1 system, suppose flow number 1 is to be allocated exclusive bandwidth. The additional capacity requirement can be calculated as follows. First, the delay for the remaining n–1 flows is:

$$T_{[n-1]} = \frac{1}{\mu C_{[n-1]} - \lambda\left(\frac{n-1}{n}\right)} \tag{7.44}$$

where $C_{[n-1]}$ denotes the capacity required for the remaining n–1 flows to maintain the same delay before separating one flow. $C_{[n-1]}$ can be calculated by solving the following equation:

$$\frac{1}{\mu\, C_{[n-1]} - \lambda\left(\frac{n-1}{n}\right)} = \frac{1}{\mu C - \lambda} \tag{7.45}$$

yielding:

$$C_{[n-1]} = C - \frac{\lambda}{n\mu} = C\left(1 - \rho\left(\frac{1}{n}\right)\right) \tag{7.46}$$

Using (7.37) and (7.46), the new total capacity $C^{[1]}$ can be calculated as:

$$C^{[1]} = C_{[n-1]} + C_{\{1\}} \tag{7.47}$$

The additional capacity $C_{\{x1\}}$ is therefore equal to:

$$C_{\{x1\}} = C^{[1]} - C = C(1-\rho) \tag{7.48}$$

For each subsequent flow *i* to be allocated exclusive bandwidth, the procedure can be repeated as follows. The delay for the remaining *n–i* flows is:

$$T_{[n-i]} = \frac{1}{\mu C_{[n-i]} - \lambda\left(\frac{n-i}{n}\right)} \tag{7.49}$$

$C_{[n-i]}$ is determined by solving:

$$\frac{1}{\mu C_{[n-i]} - \lambda\left(\frac{n-i}{n}\right)} = \frac{1}{\mu C - \lambda} \tag{7.50}$$

yielding:

$$C_{[n-i]} = C - \frac{i\lambda}{n\mu} = C\left(1-\rho\left(\frac{i}{n}\right)\right) \tag{7.51}$$

The new total capacity is $C^{[i]}$ can be calculated as:

$$C^{[i]} = C_{[n-i]} + i\,C_{\{i\}} \tag{7.52}$$

where $C^{[i]}$ denotes the new required capacity of the system after separating all flows from 1 to *i*. Note from (7.36) that $C_{\{i\}} = C_{\{1\}}, i = 2,\cdots,n$ because the formula is not restricted to a particular flow. The additional capacity $C_{\{xi\}}$ is therefore equal to:

$$C_{\{xi\}} = C^{[i]} - C^{[i-1]} = C(1-\rho) \tag{7.53}$$

Thus:

$$C_{\{xi\}} = C(1-\rho), \qquad i = 1,\cdots,n-1 \tag{7.54}$$

i.e., for each additional flow that requests an exclusive bandwidth (equal to $C_{\{1\}}$), the extra capacity that needs to be added to maintain the same delay value for the remaining flows is always the same. The additional system capacity required depends only on the system load ρ. The relation between additional capacity and system load is shown in Figure 7.2.

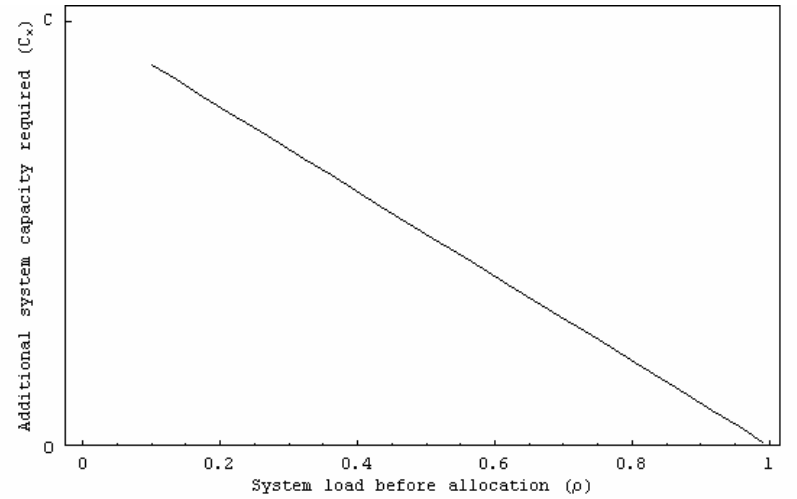

Figure 7.2: Additional system capacity required for different system loads

7.4.3 Pricing Individual Flows

In the case when some flows use a shared bandwidth and some flows request dedicated bandwidth, a suitable pricing mechanism is to use a two-part tariff, one part for the lowest QoS level and a per-unit part for the premium level [51]. In implementing the pricing scheme proposed in this chapter, let F denote the fixed part, which all flows will pay to get the average delay of T. Let R denote the tariff per unit capacity (e.g. bps). The suggested pricing scheme should calculate the price P_i for the flow i as:

$$P_i = F + R\,C_{\{xi\}} \tag{7.55}$$

It can be seen from (7.55) that flows which do not request exclusive bandwidth are charged only for the fixed part F. On the other hand, flows requesting exclusive bandwidth pay an additional rate proportional to the additional capacity required by the system to maintain an acceptable performance for all traffic flows.

7.5 Conclusions

This chapter has presented a new technique for computing the cost of providing identical channels out of a common pool of channels shared among multiple flows. Such a scheme is implemented, for example, in the IntServ system for providing a defined QoS to a set of requesting subscribers. This chapter has shown that exclusive allocation of bandwidth has a performance penalty on delay and jitter. This chapter also derived the additional capacity required to maintain the desired performance parameters and proposed a compensatory scheme that will fairly charge flows requesting exclusive bandwidth for the additional system cost. Using the process and procedure developed in this chapter, a service provider can develop an effective mechanism for establishing tariffs for IntServ customers under a wide variety of ambient conditions.

Chapter 8. Refined Assured Forwarding Framework for Differentiated Services Architecture

Abstract: This chapter presents a new framework termed the Refined Assured Forwarding (RAF) framework for improving the performance of DiffServ architecture where heterogeneous traffic flows share the same aggregate class. The new framework requires minimal modification to existing DiffServ routers. The efficiency of the new architecture in enhancing the performance of DiffServ is demonstrated by simulation results under different traffic scenarios.

8.1 Introduction

The Differentiated Services (DiffServ) architecture [42] has been designed to overcome the scalability problems of IntServ. DiffServ is a macro-level approach. In DiffServ, flows are assigned to classes and each class gets a different level of service. However, there is no differentiation between flows within the same class, except for the drop precedence. As a result, many fairness problems have been observed and discussed in published literature. These include fairness between TCP and UDP flows sharing the same class, and fairness between TCP flows with different parameters (window sizes, round-trip times) sharing the same class [12, 60]. In addition, the results of this chapter show that there is unfairness in the bandwidth sharing between UDP flows with disparate packet sizes or arrival rates within the same DiffServ class.

While different scheduling mechanisms are employed for managing the queues of different classes, flows within the same class are generally served on a FIFO basis. A large packet waiting in the queue can force many smaller packets to be delayed, which unfairly increases the overall delay of the system.

This chapter presents a new framework for improving DiffServ performance by providing a higher degree of fairness and lower average delay. The proposed approach provides additional refinement to the existing Diffserv classification scheme to separate flows with comparable characteristics into subclasses within the same DiffServ class.

8.2 DiffServ Deficiencies in Handling Heterogeneous Traffic

Despite its success as a scalable QoS architecture, DiffServ suffers from some deficiencies that have been identified in published literature. In particular, several fairness problems have been addressed in [61-65]. These problems fall under two categories: inter-class fairness and intra-class fairness [62]. Inter-class fairness refers to the fair share of resources (queue size and bandwidth) between different AF classes. It also includes the fair distribution of excess bandwidth when the network is underutilized. Some studies have shown that flows in a higher class can get worse performance than flows in lower class due to unbalanced distribution of bandwidth [63]. Intra-class fairness refers to the fair sharing of resources between different flows within the same AF class. Flow-based QoS cannot be guaranteed because DiffServ routers do not keep track of individual flows. In heterogeneous networks, different types of flows may share the same DiffServ class, e.g., TCP flows with different window sizes, a mix of TCP and UDP flows, or UDP flows with different average

packet sizes. In all of these and other scenarios, some flows will gain higher bandwidth than others, although they have the same service priority.

The main cause of intra-class fairness problems is the aggregate nature of DiffServ. This type of unfairness may be unavoidable because of the nature of DiffServ. However, some of these problems can be alleviated using efficient scheduling mechanisms.

8.3 Related Work

The fairness problems in DiffServ have been addressed in several studies. In [65], a solution is proposed for providing fair sharing of excess bandwidth for traffic flows in proportion to their in-profile rates. The proposed solution classifies flows into Aggregate Groups (AG) based on a fairness index which is calculated based on the number of arrived packets and the specified profile packets of each flow. The aggregation is done at a level that is higher than the flow level but lower than the AF class level. However, this approach uses labels instead of the DSCP field to classify flows. This adds more complexity and incompatibility problems. Also, the proposed solution creates a completely new classification mechanism rather than building on the existing AF classification. The typical number of aggregates (as used in the simulation) is relatively high compared to the number of AF classes.

In [62], two schemes are proposed for providing fairness in DiffServ. For core routers, a scheduling mechanism called Fair Weighted Round Robin (FWRR) is proposed for providing inter-class fairness between out-profile AF packets and Best Effort (BE) packets to prevent either one from unfairly monopolizing resources. This is done by dynamically allocating the bandwidth and queue limits for each class. For

edge routers, an intra-class fairness policy is designed to protect responsive flows from greedy non-responsive flows within the same class. However, the intra-class fairness is only provided at edge routers. Also, these schemes do not consider the fairness between multiple UDP flows with different packet size. It warrants further study and thus is addressed in this dissertation.

In an under-provisioned DiffServ network, flows in a lower class might get better performance than flows in a higher class because of the unbalanced distribution of bandwidth when the higher class has a larger number of flows than the lower class. This problem has been addressed in [63]. The paper proposes a technique that estimates the number of active flows in each class and uses this number to dynamically adjust the bandwidth allocated to each class. This technique requires each core router to examine the source and destination address for each packet in order to apply a hash function that estimates the number of flows, which adds more complexity to the implementation. In addition, the intra-class fairness is not considered.

The fairness between stream and non-stream flows has been studied in [66]. The paper has focused on the problem of negative interactions between TCP and UDP flows sharing the same AF class. Instead of assigning TCP and UDP flows to different classes with dynamic bandwidth allocation, the authors have proposed a new technique that assigns the TCP flows to different classes based on their arrival rate, duration and bandwidth requirements. UDP flows are then dynamically assigned to any of the four classes with admission control considering certain parameters. The problem with this approach is that it doesn't provide AF services. The criteria for

class assignment are based on application and flow characteristics and not on the SLA or customer assigned priorities.

The fairness between flows with different packet sizes has been studied in [64]. A closed loop signaling feedback technique is proposed which uses fairness information sent from the egress DiffServ router to the ingress DiffServ router to control flow admission to the DiffServ network. This approach does not involve core routers. Thus, the fairness of core routers may not be improved and no information from core routers is being used to improve the fairness at the edge routers. When there are multiple destinations for a single source, the complexity of this approach is increased limiting the scalability of this approach in large networks. In addition, signaling information is being sent using the EF class, adding an overhead to the network and reducing the throughput of premium EF traffic.

From the previous discussions, the published studies related to the area of fairness in DiffServ either do not consider the intra-class fairness problem between flows with different packet sizes, or use complex techniques to handle this problem. The motivation of this study is similar to the main purpose of DiffServ: instead of using an approach that adds a lot of overhead to achieve substantial improvement, it is more practical to add minimal overhead to achieve moderate improvement. This study shows, however, that the improvement with the small overhead can be substantial.

8.4 The Refined Assured Forwarding Framework

In Chapter 5, it has been shown that the performance of heterogeneous networks can be enhanced by dividing flows with similar characteristics into groups

rather than aggregating or fully segregating them. This approach brought in a considerable degree of refinement over the integration versus segregation studies reported previously [19, 32]. This chapter proposes a framework based on a similar concept to alleviate the intra-class unfairness within an AF DiffServ class. This approach is termed a Refined Assured Forwarding or RAF framework. The basic idea in RAF is to provide an additional layer of classification independent of the DiffServ classification criteria. Within each AF class, flows are further classified into groups based on their average packet size. This classification is done by edge routers, where flows can be tracked. For core routers, the additional classification layer is transparent, except for increasing the number of classes. In order to minimize the impact of the additional classification, the number of groups or secondary level classes should be kept minimal. This can be done, for example, by decreasing the number of AF classes and the number of drop precedence levels to compensate for the increasing number of queues managed by core routers. An overview of the RAF approach is shown in Figure 8.1. Each of the three AF classes: AF1, AF2 and AF3 are divided into four RAF subclasses: A, B, C and D. Each subclass supports two drop precedence levels.

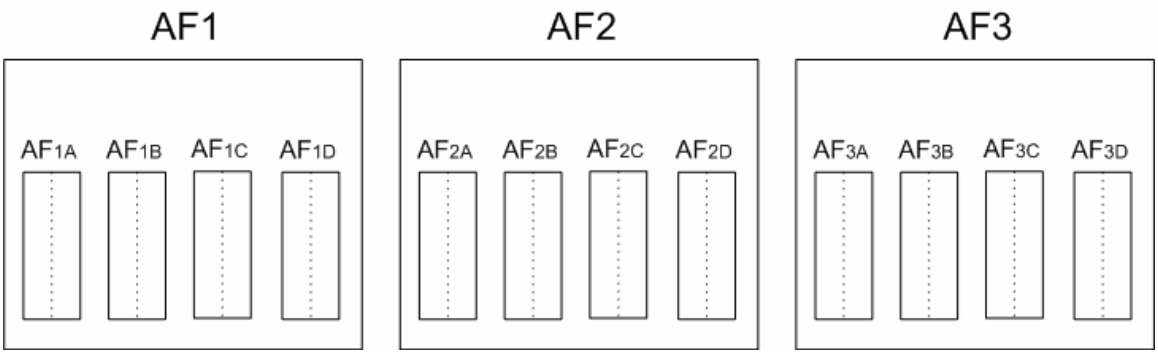

Figure 8.1: Overview of the Refined Assured Forwarding classification

8.4.1 Classification Method

In the RAF framework, classification within each AF class is based on the average packet size. From Figure 8.1, AF_1 is classified into AF_{1A}, AF_{1B} and AF_{1C}. Ideally, classification can be performed upon packet arrival by assigning it to the appropriate subclass based on the individual packet size, independent of the flow to which the packet belongs. However, AF specifications do not allow splitting a single flow between multiple classes because this will result in packets arriving out of order [67]. Therefore, it is required to make the classification on the flow level instead of on the packet level. This requires maintaining a table with flow id /average packet size pairs at the edge routers. The table can be updated by monitoring flows periodically. Typically, flows will have minimal, if any, change of their parameters during the connection lifetime. With this assumption, flow based classification can be simplified.

8.4.2 Class Assignment

The DiffServ Code Point (DSCP) is a 6-bit field. This allows up to 64 different class assignments. Two classes are already assigned to the Best Effort (BE) and Expedited Forwarding (EF) classes. This leaves 62 classes for AF. However, standard addressing conventions and addresses reserved for experimental or future use may restrict the use of all available address space. The implementation of RAF may require modifying the standard way of assigning AF DSCP values. The recommendation of this dissertation is to use only three AF classes (gold, silver and bronze) with four RAF subclasses in each class as shown in Figure 8.1. Within each RAF subclass, two drop precedence levels are defined. The total number of distinct

DSCP values required is 24. Note that the distinct DSCP values for drop precedence are considered as virtual queues, rather than physical queues. Thus, the actual number of queues managed by core routers is only 12 in this case. RAF subclasses within each AF class must be assigned DSCP values that are lower than the next AF class. This is to ensure that subclasses within a lower priority AF class don't get better treatment than higher priority AF classes. RAF subclasses are assigned by the edge router.

8.4.3 Core Router Operation

In the RAF framework, the core routers will have the flexibility to assign a separate queue for each RAF subclass or to assign all RAF subclasses to the same queue as their parent AF class. This enables the implementation of RAF in core routers with different traffic loads and across multiple DiffServ domains. Moreover, routers can choose to combine some of the RAF subclasses into one queue. Table 8.1 shows a sample database that core routers will maintain in RAF implementation. The first and second columns in Table 8.1 are for illustrative purposes and need not to be stored in the database. The third and fourth columns show suggested DSCP values to identify individual AF classes and their RAF subclasses. P1 and P2 are the two drop precedence levels within each subclass. The fifth column shows the queue number assigned by the core router to flows belonging to the corresponding DSCP. The last column is the percentage of the number of incoming packets from the corresponding RAF subclass within its parent AF class. Depending upon the implementation of the core router, this percentage can be used to decide whether to assign this subclass a separate queue or combine it with another subclass into the same queue.

AF Class	RAF Subclass	DSCP		Queue Number	% of Arriving Packets
		P1	P2		
AF1	A	001000	001001	1	20%
AF1	B	001010	001011	2	25%
AF1	C	001100	001101	3	30%
AF1	D	001110	001111	4	25%
AF2	A	010000	010001	5	30%
AF2	B	010010	010011	6	5%
AF2	C	010100	010101	6	45%
AF2	D	010110	010111	7	20%
AF3	A	011000	011001	8	10%
AF3	B	011010	011011	8	15%
AF3	C	011100	011101	8	15%
AF3	D	011110	011111	8	60%

Table 8.1: Example core router database in RAF implementation

8.4.4 Queue Management

AF specifications mandate that the implementation of AF must use an active queue management mechanism to minimize long term buffer congestion while allowing short term burstiness [67]. The commonly used queue management technique in DiffServ implementations is Random Early Detection (RED) [68]. RED uses two thresholds, minimum threshold and maximum threshold. When the queue size is below the minimum threshold, no packets are marked to be dropped. When the queue size is between the minimum threshold and the maximum threshold, packets are randomly marked with linearly increasing probability. When the queue size is beyond the maximum threshold, all packets are marked. This gradual dropping behavior helps to eliminate synchronized congestion that can happen with abrupt dropping, where all TCP connections back off and simultaneously start increasing

their window size using the Slow Start algorithm, causing the congestion scenario to repeat. In RED, packets are dropped approximately proportionally to their flow's share of bandwidth [68].

In DiffServ routers implementing RED for the AF group, each AF class has its own minimum and maximum thresholds. The proposed RAF implementation will use the queue size threshold for the parent AF class as a total. This queue size will be shared among the RAF subclasses. Several of buffer sharing techniques have been proposed in the literature. They range from Complete Partitioning (CP), where each queue is assigned a strict threshold, to Complete Sharing (CS), where all queues share the entire buffer space [69-71]. A key parameter for the success of RAF is effective management of the AF queue sharing among the RAF subclasses. This is essential in order to prevent greedy flows with large packet sizes from dominating the use of the buffer. In this dissertation, the Push-Out with Threshold (POT) technique is chosen for buffer management [70]. In POT, a threshold is set for each queue within the buffer. When the buffer is not full, it is fully shared between all queues. When the buffer is full and a new packet arrives, the threshold of its queue is checked. If the queue is beyond its threshold, the new arriving packet is dropped. If the queue is below its threshold, a packet is dropped (pushed out) from the head of the longest queue. The drop precedence must be taken into consideration. Packets with lower drop precedence should never be dropped in favor of packets with higher drop precedence.

8.4.5 Scheduling and Bandwidth Allocation

In addition to the queue management scheme, the RAF framework relies on efficient scheduling to improve the delay and throughput performance. The RAF is essentially a scheduling enhancement technique. Scheduling mechanisms control the amount of time each queue is being serviced. Examples of scheduling mechanisms are Round Robin (RR) [72], Weighted Round Robin (WRR), Deficit Round Robin (DRR) [73], Fair Queueing (FQ) and Weighted Fair Queueing (WFQ) [74, 75]. FQ attempts to emulate time division multiplexing on the packet level by using clock sequencing and time stamping. WFQ adds to FQ the ability to assign different weights to different flows [5].

In RAF, it is desired to increase the throughput and reduce the delay for all flows within the same AF class. To achieve this, the router can forward more short packets than long packets at the time of congestion. Therefore, RAF uses WFQ with weights assigned by the edge router to each subclass according to the number of flows in this subclass. The next section provides simulation results of the RAF implementation.

8.5 Simulation Results

To demonstrate the performance enhancement provided by RAF implementation, several simulation experiments have been performed. The performance is measured on three criteria: average delay per packet, packet drop rate and throughput. Figure 8.2 shows the simulation setup used in all simulation experiments. Eight source nodes are connected to a DiffServ edge router (E1). Each source has a UDP agent attached to it. One sink (null) agent is attached to the

destination node (Dest). The destination node is attached to a DiffServ edge router (E2). One DiffServ core router (Core) links the two edge routers. Simulations have been done using the Network Simulator ns-2 [76].

To study the effect of scheduling on the RAF performance, the WFQ scheduler was used on E1 and Core routers. The edge router E1 and the core router Core implemented ds/RED queue provided by Nortel DiffServ module [77]. The WFQ module was obtained from [78]. E2 router used a simple DropTail FIFO queue. The weights assigned to the RAF subclasses were proportional to the number of flows in each subclass at the edge router E1. In the core router, all subclasses were assigned equal weights, as it is assumed that DiffServ core routers do not generally keep track of individual flows. To make a fair comparison between the proposed RAF framework and the regular DiffServ AF operation, all network parameters were identical in RAF and AF simulations, except for the classification provided by RAF with weights assigned to each subclass. In the AF single class, one WFQ queue was used in both edge and core routers.

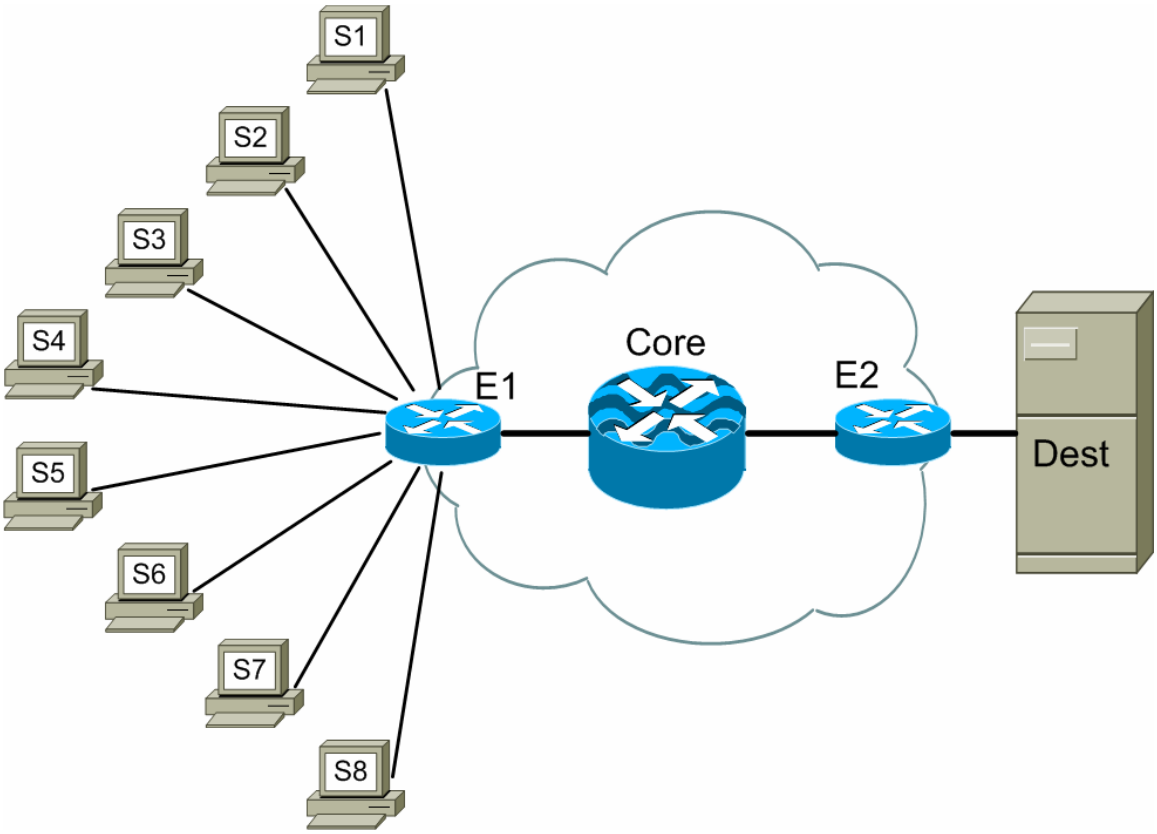

Figure 8.2: Network topology used in the simulations

One UDP agent is attached to each source (S1 to S8). Throughout the 30 seconds duration of each simulation experiment, each source generated a single flow. Table 8.2 shows the five different datasets used in all simulation experiments. In Table 8.2, C is the link capacity in Mbps, ρ is the link utilization, λ is the arrival rate in packets per second and L is the average packet size in bytes per packet. A Pareto distribution [27] was used for both interarrival and packet size distributions to produce self-similar flows [24]. All simulations were performed under two scenarios. Since the purpose of the simulation is to study the intra-class fairness performance, the standard scenario used only one AF class for all flows. In the second scenario, the four RAF subclasses have been implemented. Table 8.3 presents the classification of flows into the RAF subclasses based on their source-destination addresses.

In the results comparison below, the results obtained by the proposed RAF framework are referred to simply RAF. Similarly, the results of regular DiffServ AF implementation are referred to as AF

			Dataset 1		Dataset 2		Dataset 3		Dataset 4		Dataset 5	
src	C	ρ	λ	L	λ	L	λ	L	λ	L	λ	L
	Mbps		pkt/s	B/pkt	pkt/s	B/pkt	pkt/s	B/pkt	pkt/s	B/pkt	pkt/s	B/pkt
S1	1	0.70	1750	50	1750	50	1750	50	1750	50	1750	50
S2	1	0.70	1591	55	1591	55	1591	55	1591	55	1591	55
S3	1.5	0.75	469	300	282	500	141	1000	71	2000	47	3000
S4	1.5	0.75	427	330	266	530	157	900	75	1900	44	3200
S5	1.5	0.75	447	315	256	550	118	1200	64	2200	43	3300
S6	2	0.80	250	800	167	1200	67	3000	34	6000	23	9000
S7	2	0.80	241	830	143	1400	58	3500	31	6500	22	9500
S8	3	0.90	169	2000	85	4000	34	10000	23	15000	19	18000

Table 8.2: Input data used in the simulations

Source	Destination	AF class	RAF subclass
S1	Dest	AF1	A
S2	Dest	AF1	A
S3	Dest	AF1	B
S4	Dest	AF1	B
S5	Dest	AF1	B
S6	Dest	AF1	C
S7	Dest	AF1	C
S8	Dest	AF1	D

Table 8.3: Flow classification in the simulations

8.5.1 Delay Performance

To evaluate delay performance, the queues of E1 and the Core router (Figure 8.2) were assigned very large buffers to emulate infinite queue lengths. The E1-Core and Core-E2 links were assigned a lower bandwidth than the sum of the bandwidth of the links S1-E1 through S8-E1 to force congestion on the edge and core routers.

Table 8.4 presents the average delay per packet that was measured for each individual flow and for the combined traffic. The rows represent each of the flows. The delays with and without the use of the proposed scheme are shown in the RAF and AF columns, respectively, for each of the datasets. From the results, it can be seen that the individual delays are significantly lower for RAF than AF for smaller packet flows. For the combined traffic, the average delay has improved by about 70%. The results are graphically presented in Figure 8.3. The results clearly demonstrate the impact of the fair scheduling achieved by RAF.

Average Delay Per Packet										
Dataset	1		2		3		4		5	
Flow	RAF	AF	RAF	AF	RAF	AF	RAF	AF	RAF	AF
1	0.371	3.782	0.948	4.626	1.189	4.371	0.571	2.876	0.898	3.647
2	0.389	3.865	0.889	4.591	1.200	4.391	0.590	2.896	0.885	3.613
3	0.431	3.854	1.182	4.692	1.664	4.670	2.194	3.226	1.039	3.499
4	0.454	3.848	1.140	4.567	1.475	4.309	2.093	3.024	1.112	3.787
5	0.448	3.506	1.162	4.706	1.570	4.468	2.385	3.225	1.130	4.006
6	4.591	3.905	4.745	4.678	5.813	4.742	3.024	3.103	7.170	6.480
7	4.306	3.751	4.698	4.747	7.458	5.791	3.182	3.088	5.423	4.767
8	9.315	3.799	10.071	8.508	8.681	5.313	6.175	3.216	6.759	3.995
Combined	0.852	3.803	1.223	4.681	1.381	4.419	0.726	2.908	0.960	3.663

Table 8.4: Simulation results – delay

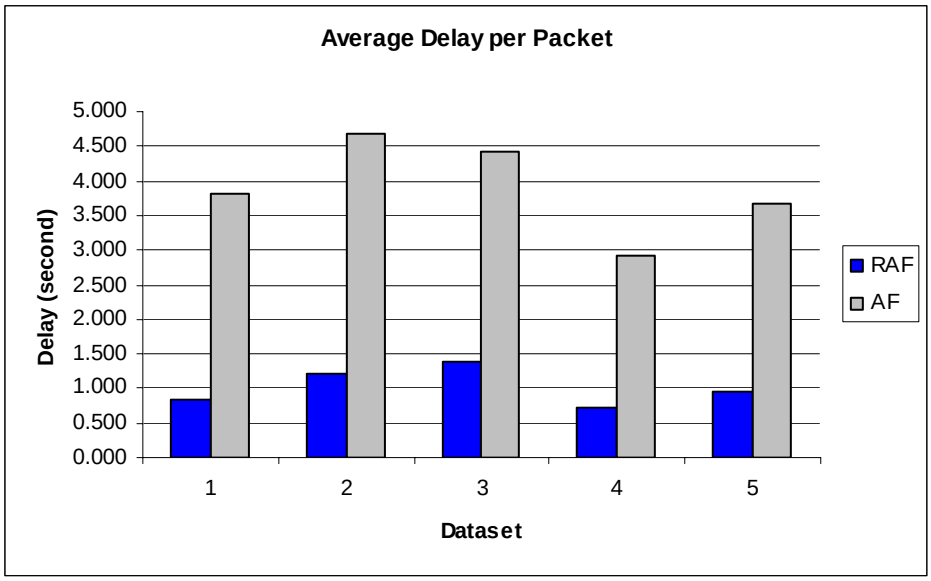

Figure 8.3: Simulation results – average delay per packet for the combined flows

Table 8.5 presents the number of received packets per flow. The results are graphically presented in Figure 8.4. Figure 8.4 clearly shows that the number of received packets is significantly higher for the RAF implementation.

Packet Received										
Dataset	1		2		3		4		5	
Flow	RAF	AF	RAF	AF	RAF	AF	RAF	AF	RAF	AF
1	51643	40333	46753	34322	50710	39373	50969	41667	50577	38907
2	47165	36442	44385	33754	43921	34508	46650	37931	46361	35850
3	12272	11058	8107	6082	3526	2656	2020	1857	1323	986
4	12772	10002	7812	5975	4435	3458	2252	2022	1324	982
5	11416	9110	7443	5621	3547	2771	1762	1527	1332	1049
6	5946	6093	3599	3785	1507	1595	805	875	445	506
7	5320	5560	2842	3064	908	908	703	746	378	463
8	2465	3945	742	1353	466	739	312	411	292	458
Total	148999	122543	121683	93956	109020	86008	105473	87036	102032	79201

Table 8.5: Simulation results – packet delivery

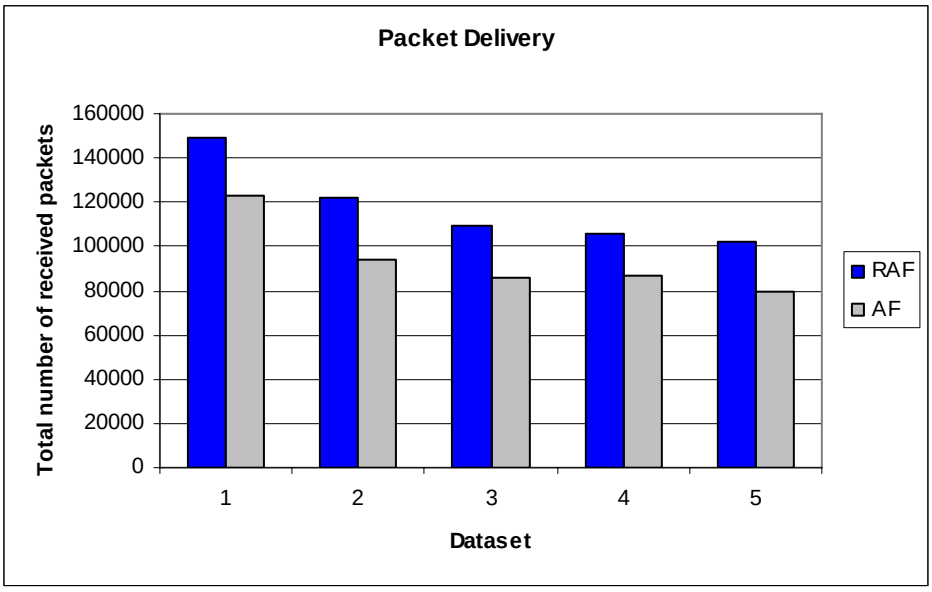

Figure 8.4: Simulation results – number of packets delivered

8.5.2 Throughput Performance

Throughput is the number of bytes received per second from each flow. The throughput performance was evaluated in the same experiments as those used for delay evaluation. Table 8.6 shows throughputs for individual flows and the total throughput for the combined traffic. Figure 8.5 presents the throughput performance of individual flows for dataset 2. While the total throughput is almost the same for RAF and AF in all datasets, it can be seen from Figure 8.5 that throughput is more evenly distributed among the flows in RAF than in AF. The proposed scheme thus illustrates not only improvements in delay performance, but also in fairness among traffic flows.

Throughput										
Dataset	1		2		3		4		5	
Flow	RAF	AF	RAF	AF	RAF	AF	RAF	AF	RAF	AF
1	86215	66628	78415	56335	81624	63092	83402	67581	83460	62923
2	87485	66795	79260	59794	80494	62034	83959	67787	82872	64261
3	115980	104755	124933	93996	116270	90983	132107	119355	112289	84395
4	134402	104828	134854	102357	124133	95983	132999	120973	130780	95957
5	114892	89634	129092	98421	131967	102980	132825	116659	127018	104660
6	154975	162093	135946	143104	149452	157301	147649	161186	155115	179894
7	144656	152020	130066	138546	128141	128141	130874	141239	131234	151828
8	149535	241482	133276	252615	141002	252558	144252	194256	148998	228352
Total	988139	988235	945842	945170	953084	953071	988068	989036	971765	972270

Table 8.6: Simulation results - throughput

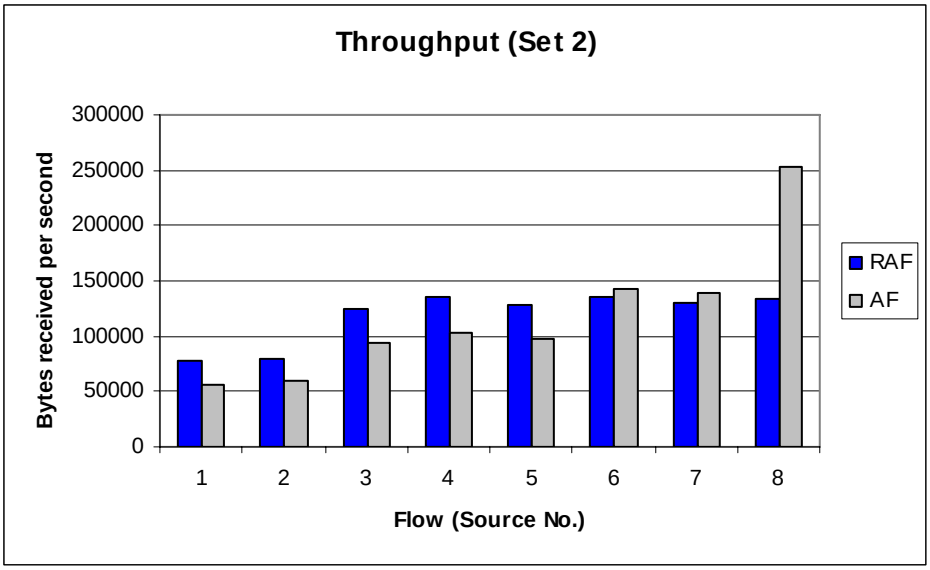

Figure 8.5: Simulation results – throughput for individual flows in dataset 2

8.5.3 Packet Loss

To evaluate the packet loss performance, buffer sizes were decreased in the E1 and Core routers. All other parameters were kept identical to the other simulation experiments, including the RED minimum and maximum thresholds. Table 8.7 and

Figure 8.6 show the number of dropped packets for individual flows and for the combined traffic, respectively. From Table 8.7, it can be seen that, without implementing RAF, the number of small packets dropped is much higher than the number of large packets dropped. In addition, the RAF implementation improved the total packet loss in the combined traffic, as shown in Figure 8.6.

Packet Loss										
Dataset	1		2		3		4		5	
Flow	RAF	AF	RAF	AF	RAF	AF	RAF	AF	RAF	AF
1	1137	15427	5248	14929	4973	15868	4051	15640	4744	18576
2	950	13435	4988	14091	4699	14308	3846	13910	4069	16167
3	682	3668	1199	2250	452	1204	126	580	126	442
4	443	3163	1206	1906	557	1265	98	633	90	483
5	601	3635	1047	2011	349	940	103	480	111	442
6	2629	1660	1353	1186	449	481	46	256	16	259
7	2401	1763	1165	1162	282	503	56	240	19	82
8	2805	1416	1138	667	87	228	18	205	11	212
Total	11648	44167	17344	38202	11848	34797	8344	31944	9186	36663

Table 8.7: Simulation results – packet loss

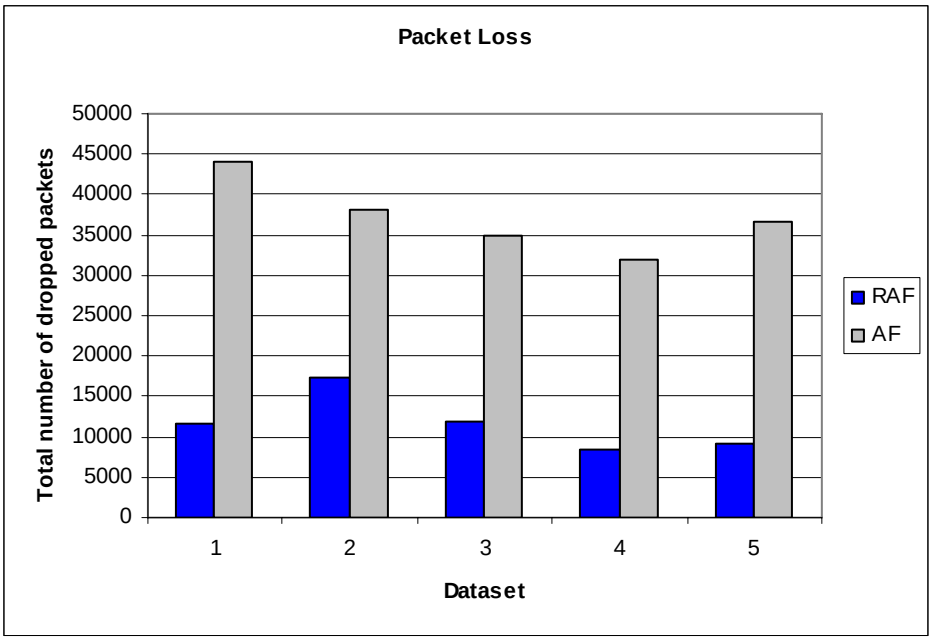

Figure 8.6: Simulation results – packet loss comparison

8.5.4 Discussion of the Results

The input flows used in the simulation had disparate packet sizes. Without the isolation provided by the RAF framework, flows with large packet sizes, such as S8, dominated the use of the buffer and bandwidth. This caused longer waiting in the queue and less delivery rate for smaller packet flows. When the buffer size was limited, more small packets were dropped in favor of large packets. By using the RAF classification, flows with small packets got a fairer share of resources. With more weight given to subclasses with more flows, the fairness was further enhanced. As a result, the delay, throughput and packet loss of flows with small packets were improved, which also improved the performance of the combined traffic.

8.6 Conclusions

This chapter has introduced the Refined Assured Forwarding (RAF) framework, which adds an additional layer of classification to flows within the DiffServ AF classes based on the average packet size in each flow. RAF uses Weighted Fair Queueing and assigns weights to subclasses proportional to the number of active flows in each subclass at the edge router. Core routers use Weighted Fair Queueing (WFQ) with equal weights and may combine RAF subclasses or disregard the RAF classification as needed. The performance of RAF in terms of average delay, throughput and packet loss has been demonstrated by means of simulations. The simulation results have shown significant improvements for both individual flows and combined traffic.

Chapter 9. Conclusions and Future Work

This dissertation has studied the performance of heterogeneous large scale communication networks. As shared resources, packet switched networks typically carry traffic from various applications with diverse characteristics and, sometimes, conflicting requirements. This dissertation has built on the notion that when different flows or classes of traffic have disparate characteristics, it is more efficient to isolate or segregate those flows than to combine or integrate them. While this was previously proven for only limited scenarios under classical queueing models, this dissertation has presented a new approach as well as considerably weakened the assumption of traffic distribution by including self-similarity and general distribution of service time. Numerous recent studies have shown that Poisson models are inadequate in describing network traffic in many network environments.

In the first part of this dissertation, the segregation versus integration question has been extended to include queues with an arbitrary number of flows. Different queueing models have been evaluated, including M/G/1 and G/G/1 queues. Evaluation of these models entailed in depth research in mathematical and engineering literature. A new queue management technique, termed Hybrid Integration, has been presented. In this approach, segregation is done at a group level rather than on a flow level. Flows with similar parameters are grouped together into the same channel. This approach is more appropriate for large systems where complete segregation is not possible. The design of this technique mandated defining the criteria upon which grouping is done to optimize the average queueing delay for

the system. It was also required to implement additional optimization techniques for reducing the computational complexity of grouping analysis. Analytical and simulation studies have both shown that the new Hybrid Integration approach is capable of reducing the average queueing delay compared to both the segregated or integrated approaches. At worst, the Hybrid approach matches the lowest delay of either the segregation or integration approaches.

The second part of this dissertation considered networks implementing Quality of Service (QoS) mechanisms, particularly Integrated Services (IntServ) and Differentiated Services (DiffServ). IntServ provides strict QoS assurance by providing resource reservation for each flow. Besides raising scalability issues, bandwidth reservation brings the issues of accounting and pricing of service. In this dissertation, the segregation versus integration analysis enabled introducing a new pricing model that reflects the actual bandwidth usage per flow. It has been shown that, in homogeneous traffic environments, each additional flow that requires a reserved bandwidth consumes more than simply its equal share of bandwidth had the bandwidth been equally split between flows. Additionally, a performance penalty in delay and jitter is imposed on other flows which share the remaining bandwidth that requires increasing the allocated bandwidth for these flows. The developed pricing model takes this required bandwidth increase into consideration when charging flows requesting reservation.

DiffServ was designed as a scalable alternative to IntServ. DiffServ operates on the aggregate or class level and therefore it does not differentiate between individual flows. Each packet is assigned to a particular class when entering the edge

of the network. Core routers provide differential treatment for each class without identifying which flow the packet belongs to. Despite its success, many studies have shown that DiffServ has some fairness problems when flows within the same aggregate class have disparate characteristics. For example, TCP/UDP flows or short-packet/long-packet flows. To address this problem, this dissertation has proposed a new Refined Assured Forwarding (RAF) framework. This framework provides a second level of classification to the existing DiffServ Assured Forwarding (AF) classification. AF provides four classes and three drop precedence levels. The class assignment in AF is typically based on a Service Level Agreement (SLA). The new RAF framework further classifies flows within a single AF class into multiple subclasses based on the average packet size of each flow. Inspired by the Hybrid Integration approach, the RAF framework defines a limited number of groups and assigns a range of packet sizes for each group according to the network traffic statistics. In order to improve the fairness of queue and bandwidth sharing, RAF utilizes the Weighted Fair Queueing (WFQ) scheduling mechanism, which emulates the behavior of time division multiplexing with ability to assign different weights to different flows. The performance enhancement of the RAF framework has been evaluated by simulation on different performance criteria, including delay, throughput and packet loss. Simulation results have shown that, using the RAF classification, flows with small packets got a fairer share of resources. By implementing WFQ with higher weights given to subclasses with larger number of flows, the fairness was further enhanced. As a result, the delay, throughput and packet loss of flows with

small packets were improved, which also improved the overall performance of the system.

Future work would potentially include a further generalization of the results presented in this dissertation from different standpoints. A closed form solution for the optimum channel capacity allocation for M/G/1 and G/G/1 queueing systems containing an arbitrary number of channels would improve the understanding of the channel assignment problem in a generalized manner. Mathematical models are also needed for accurately calculating the delay for M/G/1 and G/G/1 queueing models with heavy-tailed distributions having non-finite variance. More information on this topic is available at [20, 37]. Further, the availability of a detailed analysis of the resource cost for M/G/1 and G/G/1 queues will extend the results for IntServ presented in this dissertation to include the self-similar traffic. The Refined Assured Forwarding framework developed in this dissertation can be made more efficient by defining standard packet size ranges for the RAF subclasses based on statistics of the different types of traffic that traverse the Internet. It will also be possible to obtain more precise simulation results using simulation tools that enable fine tuning of class assignment, buffer allocation and scheduling methods of the DiffServ model.

References

[1] W. Stallings, *Data and Computer Communications*. New Jersey, USA: Prentice-Hall, 1997.

[2] S. V. Kartalopoulos, *DWDM Networks, Devices and Technology*. New Jersey, USA: John Wiley & Sons, 2003.

[3] P. K. Verma, *Performance Estimation of Computer Communication Networks*. Maryland, USA: Computer Science Press, 1989.

[4] P. M. Fernandez, "Circuit Switching in the Internet," PhD Thesis, Stanford University, June 2003.

[5] G. Armitage, *Quality of Service in IP Networks*. Indiana, USA: Macmillan Technical Publishing, 2000.

[6] S. Jha and M. Hassan, *Engineering Internet QoS*. London, UK: Artech House, 2002.

[7] Z. Wang, *Internet QoS Architectures and Mechanisms for Quality of Service*. California, USA: Morgan Kuffmann Publishers, 2001.

[8] W. C. Hardy, *QoS Measurement and Evaluation of Telecommunications Quality of Service*. West Sussex, England: John Wiley & Sons, 2001.

[9] M. Morrow, V. Sharma, T. D. Nadeau, and L. Andersson, "Challenges in Enabling Interprovider Service Quality in the Internet," in *IEEE Communications Magazine*, vol. 43, 2005, pp. 110-111.

[10] J. Walrand and P. Varaiya, *High-Performance Communication Networks*, Second ed. California, USA: Morgan Kufmann Publishers, 2000.

[11] W. Stallings, *High Speed Networks and Internets*. New Jersey, USA: Prentice-Hall, 2002.

[12] C. Kim, Y. Kim, and D. Montgomery, "Fairness-Guaranteed per-Class-Type Queueing and Hierarchical Packet Scheduling for DiffServ-aware-MPLS Network," IEEE Globecom 2004, Dallas, TX, USA, December, 2004, pp. 1718-1722.

[13] R. Elliott, "A Measure of Fairness of Service for Scheduling Algorithms in Multiuser Systems," Proceedings of the 2002 IEEE Canadian Conference on Electrical and Computer Engineering, 2002, pp. 1583-1588 vol.3.

[14] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*. New York, USA: John Wiley & Sons, 2000.

[15] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic," SIGCOMM '93, September, 1993.

[16] L. Kleinrock, *Queueing Systems - Volume I: Theory*. New York, USA: John Wiley & Sons, 1975.

[17] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "Internet Traffic Tends Toward Poisson and Independent as the Load Increases," Nonlinear Estimation and Classification, New York, USA, 2002.

[18] L. Kleinrock, *Communication Nets*. New York, USA: Dover Publications, 1964.

[19] H. R. Rudin, "On Economies of Scale and Integration of Services in Certain Queued Information Transmission Systems," *IEEE Transactions on Communications*, vol. 20, No. 5, 1972, pp. 991-995.

[20] O. J. Boxma and J. W. Cohen, "The M/G/1 Queue with Heavy-Tailed Service Time Distribution," *IEEE Journal in Selected Areas in Communications*, vol. 16, No. 5, 1998, pp. 749-763.

[21] J. F. C. Kingman, "Inequailites in the Theory of Queues," *Journal of the Royal Statistical Society*, vol. 32, series B, No. 1, 1970, pp. 102-110.

[22] L. Kleinrock, *Queueing Systems - Volume II: Computer Applications*. New York, USA: John Wiley & Sons, 1976.

[23] E. W. Weisstein, "Cantor Set", Wolfram Web Resource, mathworld.wolfram.com/CantorSet.html.

[24] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, vol. 2, No. 1, 1994, pp. 1-15.

[25] V. Paxon and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, No. 3, 1995, pp. 226-244.

[26] M. E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Transactions on Networking*, vol. 5, No. 6, 1997, pp. 835-846.

[27] V. Paxon, "Empirically Derived Analytic Models of Wide-Area TCP Connections," *IEEE/ACM Transactions on Networking*, vol. 2, No. 4, 1994, pp. 316-336.

[28] E. W. Weisstein, "Pareto Distribution", Wolfram Web Resource, mathworld.wolfram.com/ParetoDistribution.html.

[29] E. W. Weisstein, "Log Normal Distribution", Wolfram Web Resource, mathworld.wolfram.com/LogNormalDistribution.html.

[30] M. H. Dahshan and P. K. Verma, "Performance Enhancement by Segregation and Hybrid Integration in General Queueing Networks," International Symposium on Performance Evaluation of Computer and Telecommunication Systems – SPECTS'05, Philadelphia, PA, USA, July 24-28, 2005, pp. 143-148.

[31] M. H. Dahshan and P. K. Verma, "Performance Enhancement of Heavy Tailed Queueing Systems using a Hybrid Integration Approach," IEEE Global Telecommunications Conference – GLOBECOM'05, St. Louis, MO, USA, Nov 28-Dec 2, 2005.

[32] P. K. Verma and A. M. Rybzcynski, "The Economics of Segregated and Integrated Systems in Data Communication with Geometrically Distributed Message Length," *IEEE Transactions on Communications*, 1974, pp. 1844-1848.

[33] C. F. Su and G. d. Veciana, "Statistical Multiplexing and Mix-Dependent Alternative Routing in Multiservice VP Networks," *IEEE/ACM Transactions on Networking*, vol. 8, No. 1, 2000, pp. 99-108.

[34] Y. H. Kim and C. K. Un, "Performance Analysis of Statistical Multiplexing for Heterogeneous Bursty Traffic in ATM Network," *IEEE Transactions on Communications*, vol. 42, No. 2/3/4, 1994, pp. 745-753.

[35] B. N. W. Ma and J. W. Mark, "Performance Analysis of Burst Switching for Integrated Voice/Data Services," *IEEE Transactions on Communications*, vol. 36, No. 3, 1988, pp. 282-297.

[36] H. Eberle and N. Gura, "Separated High-bandwidth and Low-latency Communication in the Cluster Interconnect Clint," Proceedings of the 2002 ACM/IEEE conference on Supercomputing, Baltimore, Maryland, 2002, pp. 1-12.

[37] O. J. Boxma and J. W. Cohen, "Heavy-Traffic Analysis for the GI/G/1 Queue with Heavy-Tailed Distributions," *Probability, Networks and Algorithms, PNA-R9710*, 1997.

[38] S. Kumar and P. R. Kumar, "Performance Bounds for Queueing Networks and Scheduling Policies," *IEEE Transactions on Automatic Control*, vol. 39, No. 8, 1994, pp. 1600-1611.

[39] P. K. Pollett, "Residual Life Approximations in General Queueing Networks," *Elektronische Informationsverarbeitung und Kybernetik*, vol. 2, 1984, pp. 41-54.

[40] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.

[41] U. Payer, "DiffServ, IntServ, MPLS", www.iaik.tu-graz.ac.at/teaching.

[42] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December 1998.

[43] G. Camarillo, "Routing architecture in DiffServ MPLS networks", www.netlab.hut.fi/opetus/s38130/k00/Papers/Topic1-architecture.pdf.

[44] ISI, "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, January 1981.

[45] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, January 2001.

[46] N. Rouhana and E. Horlait, "Differentiated Services and Integrated Services Use of MPLS," Fifth IEEE Symposium on Computers and Communications - ISCC'00, Antibes-Juan les Pins, France, July 3-6, 2000, pp. 194-199.

[47] G. Kotsis, "Communication using Continuous Media", www.tk.uni-linz.ac.at/teaching.

[48] M. H. Dahshan and P. K. Verma, "Pricing for Quality of Service in High Speed Packet Switched Networks," IEEE Workshop on High Performance Switching and Routing - HPSR'06, Poznan, Poland, June 7-9, 2006.

[49] L. Zhen, L. Wynter, and C. Xia, "Usage-Based Versus Flat Pricing for E-Business Services with Differentiated QoS," IEEE International Conference on E-Commerce - CEC '03, 2003, pp. 355-362.

[50] P. C. Fishburn and A. M. Odlyzko, "Dynamic Behavior of Differential Pricing and Quality of Service Options for the Internet," First International Conference on Information and Computation Economies, Charleston, South Carolina, United States, 1998, pp. 128-139.

[51] S. SeungJae and M. B. H. Weiss, "Simulation Analysis of QoS Enabled Internet Pricing Strategies: Flat Rate vs. Two-Part Tariff," 36th Annual Hawaii International Conference on System Sciences, 2003.

[52] Z. Guanxiang, L. Yan, Y. Zongkai, and C. Wenqing, "Auction-Based Admission Control and Pricing for Priority Services," 29th Annual IEEE International Conference on Local Computer Networks, 2004, pp. 398-399.

[53] M. Mandjes, "Pricing Strategies under Heterogeneous Service Requirements," Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM'03, San Francisco, CA, USA, March 30 - April 3, 2003, pp. 1210-1220.

[54] J. K. MacKie-Mason and H. R. Varian, "Pricing Congestible Network Resources," *IEEE Journal on Selected Areas in Communications*, vol. 13, No. 7, 1995, pp. 1141-1149.

[55] M. Yuksel and S. Kalyanaraman, "Pricing Granularity for Congestion-Sensitive Pricing," Eighth IEEE International Symposium on Computers and Communications, 2003, pp. 169-174.

[56] V. S. Frost and B. Melamed, "Traffic Modeling for Telecommunications Networks," *IEEE Communications Magazine*, vol. 32, No. 3, 1994, pp. 70-81.

[57] B. Duysburgh, S. Vanhastel, B. De Vreese, C. Petrisor, and P. Demeester, "On the Influence of Best-Effort Network Conditions on the Perceived Speech Quality of VoIP Connections,"

Tenth International Conference on Computer Communications and Networks, 2001, pp. 334-339.

[58] N. Davies, J. Holyer, and P. Thompson, "End-to-end Management of Mixed Applications Across Networks," IEEE Workshop on Internet Applications, 1999, pp. 12-19.

[59] M. Karol, P. Krishnan, and J. J. Li, "enProtect: Enterprise-Based Network Protection and Performance Improvement," *Avaya Labs Research - Technical Report*, 2002.

[60] L. Dong and M. Robert, "Dynamics of Random Early Detection," ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Cannes, France, 1997, pp. 127-137.

[61] H. T. Phan and D. B. Hoang, "FICC-DiffServ: A New QoS Architecture Supporting Resources Discovery, Admission and Congestion Controls," Third International Conference on Information Technology and Applications - ICITA'05, Sydney, Australia, July 4-7, 2005.

[62] Y. Sungwon, D. Xidong, G. Kesidis, and C. R. Das, "Providing Fairness in DiffServ Architecture," IEEE Global Telecommunications Conference, GLOBECOM '02, 2002, pp. 1435-1439.

[63] J.-S. Li and C.-S. Mao, "Providing Flow-based Proportional Differentiated Services in Class-based DiffServ Routers," *IEE Proceedings on Communications*, vol. 151, No. 1, 2004, pp. 82-88.

[64] M. Li and D. B. Hoang, "Resource Discovery and Fair Intelligent Admission Control over Differentiated Services Networks for Variable-Length Packets," IEEE Tenth Asia-Pacific Conference on Communications, Beijing, China, August, 2004, pp. 499-503.

[65] A. Sang, H. Zhu, and S. Li, "Weighted Fairness Guarantee for Scalable DiffServ Assured Forwarding," IEEE International Conference on Communications - ICC, June, 2001, pp. 2365-2369.

[66] K. Yasukawa, K.-i. Baba, and K. Yamaoka, "Class Assigning Management for Stream Flows Considering Characteristics of Non-stream Flow Classes," Telecommunications Network Strategy and Planning Symposium. NETWORKS 2004, 11th International, June, 2004, pp. 75- 80.

[67] J. Heinanen, T. Finland, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, June 1999.

[68] S. Floyd and V. Jacobson, "Random Early Detection for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, No. 4, 1993, pp. 397-413.

[69] F. Kamoun and L. Kleinrock, "Analysis of Shared Finite Storage in a Computer Network Node Environment Under General Traffic Conditions," *IEEE Transactions on Communications*, vol. 28, No. 7, 1980, pp. 992-1003.

[70] M. Markaki and I. S. Venieris, "A Novel Buffer Management Scheme for CBQ-Based IP Routers in a Combined IntServ and DiffServ Architecture," Fifth IEEE Symposium on Computers and Communications - ISCC 2000, Antibes-Juan les Pins, France, 2000, pp. 347-352.

[71] F. Agharebparast and V. C. M. Leung, "On the Deployment of RED on Shared-Memory Buffers," *IEEE Communications Letters*, vol. 6, No. 10, 2002, pp. 458-460.

[72] E. Hahne and R. Gallager, "Round Robin Scheduling for Fair Flow Control in Data Communication Networks," IEEE International Conference on Communications, June, 1986.

[73] M. Shreedhar and G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin," *IEEE/ACM Transactions on Networking*, vol. 4, No. 3, 1996, pp. 375-385.

[74] A. Demers, S. Keshav, and S. Shenker, "Analysis And Simulation of a Fair Queueing Algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 19, No. 4, 1989, pp. 1-12.

[75] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single-Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, No. 3, 1993, pp. 344-357.

[76] The Network Simulator ns-2", www.isi.edu/nsnam/ns.

[77] J. Ethridge, M. Baines, and F. Shallwani, "A Network Simulator Differentiated Services Implementation," *Open IP, Nortel Networks*, 2001.

[78] A. Mrkaic, "Porting a WFQ Scheduler into ns-2's DiffServ Environment," *Student Thesis SA-2001.37*, 2001.